AFIT/GSO/ENG/92D-03

# A TOPOSCOPIC INVESTIGATION OF BRAIN ELECTRICAL ACTIVITY INDUCED BY MOTION SICKNESS

THESIS

Dwight Andrew Roblyer
Captain, USAF

AFIT/GSO/ENG/92D-03

# Thesis Approval

| | | |
|---|---|---|
| *STUDENT:* | Capt. Dwight A. Roblyer | *CLASS:* GSO-92D |
| *THESIS TITLE:* | A Toposcopic Investigation of Brain Electrical Activity Induced by Motion Sickness | |
| *DEFENSE DATE:* | November 12, 1992 | |

| *COMMITTEE:* | *NAME/DEPARTMENT* | *SIGNATURE* |
|---|---|---|
| Advisor | Steven K. Rogers, Maj, USAF<br>Professor of Electrical Engineering | [illegible] |
| Reader | Thomas S. Kelso, Maj, USAF<br>Assistant Professor of Space Operations | [illegible] |
| Reader | Matthew Kabrisky<br>Professor Emeritus of Electrical Engineering | [illegible] |

AFIT/GSO/ENG/92D-03

# A TOPOSCOPIC INVESTIGATION OF BRAIN ELECTRICAL ACTIVITY INDUCED BY MOTION SICKNESS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Space Operations

Dwight Andrew Roblyer, B.A.
Captain, USAF

December, 1992

# *Preface*

I began this research as a repeat participant in the "barf chair" last fall before funds had been completely exhausted for data collection. It certainly wasn't glamorous, but I had become immediately interested upon my entering AFIT in the leading-edge research of the motion sickness research group. I was saddened by Dr. Bill Chelen's departure as a result of the budget reductions, but was glad to have · thesis topic in this fascinating area and the support of the group, who took me onboard despite the throngs of other students at their office doors.

I recently read that one person can learn when left on his own. But that person must be in the company of others in order to learn well. This statement summarizes the thesis experience for me. I am glad this page provides a forum to thank those who accompanied me.

My deepest thanks go to my wife, Kathy, and two children, Andrew and Patrick. I have appreciated their endless understanding and support, but especially the way that their presence and love were frequent reminders of the immeasurable value of my family. Maj Steve Rogers, LtCol T.S. Kelso, and Dr. Matthew K?' ısky, my committee members, provided the right mixture of guidance and expectation. They were mentors, not taskmasters, for which I am most grateful. Their enthusiasm was catching and sustaining. Capt Chuck Wright started me on the way by graciously offering his program and expertise in answer to my initial question of "How?" Capts Dave Swanson and Art Earl were dependable allies and continuous sources of workarounds and shortcuts which kept bearable the tedium of programming. Capt Ken Langford and Mr. Ron Milliron were my "friends in high places" that kept the computer workstations willing to work with me.

I thoroughly enjoyed this epic journey into the world of motion sickness, EEGs, .d computer programming. Furthermore, I will never experience motion sickness in the same way again.

I hope that in this search for pattern, Grey Walter would have seen the essence of science. But moreover, I hope that in the pursuit of science, more people will recognize the work of a loving and living God.

Dwight Andrew Roblyer

*Table of Contents*

Page

## *List of Figures*

## *List of Tables*

AFIT/GSO/ENG/92D-03

## *Abstract*

A toposcope was constructed as a new tool to study signal frequency and phase relationships in electroencephalogram (EEG) records collected while subjects were experiencing motion sickness. This new tool, named TOPOS, is a software-based, multi-featured version of Grey Walter and Harold Shipton's device which they first produced in the late 1940s. The TOPOS graphical display permits the study of instantaneous frequency relationships between the input channels and a reference signal of fixed or varying frequency. TOPOS adds a correlation grid to aid observers in detecting channel-to-reference correlation levels. Users can also vary several display parameters via menus to optimize the analysis environment. Sinusoidal test inputs of known frequency produced recognizable and predictable patterns on the TOPOS display, depending on the existing channel-to-reference frequency relationships. Motion-sickness-affected EEG was input to TOPOS in order to study the correlation between the displays of each channel and four separate references: a 1.5-Hz sinusoid, and three channels of the EEG itself. Rapidly varying correlations were observed in each case.

# A TOPOSCOPIC INVESTIGATION OF BRAIN ELECTRICAL ACTIVITY INDUCED BY MOTION SICKNESS

# *I. Introduction*

## *1.1 Background: Motion Sickness on Earth and in Space*

Motion sickness is a common companion of people traveling by land, sea, or air (33:29–30). These sufferers find that the slowly rolling road, choppy seas, or air turbulence can quickly bring on headaches, sweating, stomach discomfort, nausea, and vomiting—all symptoms of motion sickness (42:A1).[1] Victims often either take medication which makes them drowsy and the symptoms less pronounced, or stalwartly suffer until the motion stops, they adapt, or emesis brings some relief. No cure exists, and the exact etiology (cause) of the symptoms is poorly understood (28:218). While wretchedly uncomfortable and annoying, terrestrial motion sickness is relatively harmless, causing delays in travel or task schedules but posing few threats to human life (33:73).[2]

But what can be done when a person experiences motion sickness symptoms in space, where task delays carry very expensive price tags and compromise of a crewmember's attentiveness and fitness could risk his health, or even his life? Even after 30 years of research, this question remains without a wholly satisfactory answer. Moreover, it is an important question, because upon entry into orbit the first

[1]Motion sickness is really not a "sickness" at all, but was so labeled because of the symptoms it produces (33:28). Dhenin calls it a "normal response to an abnormal environment" (9:472).

[2]One significant exception is the case of military forces. Motion sickness is a significant problem for soldiers being transported into battle, explaining why the military has sponsored much of the search for the causes and cures of motion sickness (42:A1)(33:31-33).

physiologic change experienced by up to 67 percent of US astronauts is space motion sickness (SMS), also known as space adaptation syndrome (SAS) (15:1).

Space adaptation syndrome has been called by Vanderploeg, *et al.* the most clinically significant medical phenomenon during the first several days of spaceflight (8:1185). During that period, space travelers experience some symptoms common to terrestrial motion sickness and some unique to SMS. An affected person may develop a hypersensitivity to motion or experience illusions of being constantly upside down or pitched forward (42:A2). The astronaut will almost always have no appetite and may have repeated episodes of brief, but violent, emesis (42:A2). Nearly one-third of US astronauts have reported at least one episode of vomiting while in orbit (25:101). While the sufferer rarely sweats or shows pallor, he will almost certainly experience a headache, malaise, lack of initiative, and irritability (42:A3). These symptoms not only vary between individuals, but between separate trips into orbit by the same person (25:103). However it presents itself, the SMS syndrome lasts for an average of 3 to 5 days, until the victim adapts to the weightless environment (25:20).

In 1982, the Air Force Institute of Technology began research into causes and treatments of SMS. Recently, this has focused on the brain electrical activity which appears as motion sickness symptoms develop. The AFIT research team found this activity formed a pattern which was repeated among different subjects, and has followed this lead in search of insights into the causes and effective treatment of motion sickness. Banducci and Vogen began the process of analyzing these patterns by separating them into their component frequencies using spectral analysis and then graphically mapping the results (2:ix)(44:vii). Their methods averaged data across time and based results on interpolated data. Because of this methodology, there still remains questions concerning the origin and instantaneous propagation of this abnormal activity in the brain. These issues can be further investigated using a different tool—the toposcope.

## *1.2 Problem and Scope*

The problem is to construct a toposcope as a new tool, test its use and flexibility, and then interface it with motion-sickness-affected EEG. The device will be computer-based and coded in the C programming language for implementation on a Silicon Graphic workstation. This brain mapping approach should provide a continuous, graphical display which can focus on brain activity in very-low-frequency regions without the loss of time resolution resulting from spectral analysis. It will be able to derive its frequency selection from a fixed frequency source or the signal in a single EEG channel. The tool's performance will be characterized by inputting known-frequency signals. The toposcope will then be interfaced to a motion-sickness-affected EEG record.

## *1.3 Role of Pattern Recognition*

The pursuit of information about, and classification of, spatial and temporal patterns is the focus of the field of pattern recognition (43:5). In this thesis, the principles of this field will be applied to a motion-sickness-affected EEG record to search for features or invariant attributes of SMS brain activity patterns. This research effort will involve a partnership between human and machine in performing this task. The machine—the toposcope—will possess no automatic recognition capability, as is commonly associated with machines in the field of pattern recognition. Instead, the toposcope will exhibit the features of the pattern which are thought to be the best for recognition: highlighting the frequency and spatial relationships of the EEG signals. Furthermore, the toposcope will further aid recognition by tuning its entire display to the predominant frequency of a selected channel. Therefore, by extracting significant features of the EEG data from a background of irrelevant detail, a human can approach a much simpler task of recognizing any pattern present and further defining the pattern class encompassing motion sickness. Grey Walter, one of the toposcope's creators and a pioneer in brain activity research, wrote that in the seeking of pattern lies the essence of science (47:69).

## *1.4 Assumptions*

The are several key assumptions for this research regarding data collection and analysis. Because of the constraint on data availability as explained in the first assumption, several of those that follow are similar to the assumptions proposed by Vogen.

1. The only EEG records available for analysis are those collected during 1991. *Justification:* Air Force sources could no longer fund AFIT's motion sickness research after September 1991 due to budgetary cutbacks. *Limiting Effects:* No additional data can be collected for the purpose of pattern validation or in the case of data loss or corruption.

2. All the equipment used in the collection of EEG data was properly calibrated and functioning as designed. *Justification:* Data were collected prior to this research effort and Vogen documented his calibration and functional checks of equipment (44:31). *Limiting Effects:* If calibration or other problems are found in the data, efforts are restricted to finding workarounds or eliminating the affected data set(s).

3. The EEG data subject to analysis were relatively artifact free and any artifact which was undetected was insignificant to the results. *Justification:* Data were collected prior to this research effort and artifacts were assessed as minimal (44:4). *Limiting Effects:* If excessive artifacts are discovered in the data, efforts are restricted to finding workarounds or eliminating the affected data set(s).

4. The research problem is one of pattern recognition. A pattern is postulated to exist, but finding and displaying the best feature of that pattern will reveal the most information about its behavior. *Justification:* 1) Brain activity patterns

are the basis of the entire field of brain topographic mapping. 2) Vogen reported a standard pattern during motion sickness. *Limiting Effects:* Focuses the research on the qualitative detection of pattern.

5. The physiologic changes, including EEG, which occur as motion sickness develops, correlate to the subjective level of sickness reported by the subject. *Justification:* In 1987, Drylie, Fix, and Gaudreault developed an equation which correlates these two groups of data and trained a neural network to predict subjective levels of sickness using physiological data, according to Banducci (2:49). *Limiting Effects:* None. Rather, it permits the accurate labeling of EEG states with the current level of sickness.

6. The results will be representative of those which would be obtained from the healthy, normal population experiencing motion sickness symptoms. *Justification:* Samples were drawn from 10 healthy, normal volunteer Air Force officers ranging from 26 to 41 years of age (18:1153). *Limiting Effects:* None. Rather, it forms a portion of the basis for further extrapolating the results to cases of space motion sickness among astronauts.

## *1.5 Format of Thesis*

The next two chapters provide background information for this research. Chapter II frames the space motion sickness problem and emphasizes the role of the EEG in potential solutions. Chapter III then overviews three brain mapping techniques used in deciphering and understanding the complexities of the EEG. Chapter IV overviews the approach used in each stage of the thesis and explains the relationships between the AFIT display and that of Walter and Shipton (45:283). Chapter V reports on the design and coding of the AFIT toposcope. The results from analyzing test signals and motion-sickness-affected EEG records using this tool appear in Chapter VI, followed by conclusions in Chapter VII.

# II. SMS: A Summary of Current Knowledge

## 2.1 Why SMS is Significant

The impact of SMS on space operations is disputed by some. Although acknowledging the syndrome's physiological effects, Thornton, *et al.* state,

> While it is obvious that a person is *hors de combat*[1] during vomiting, this is brief. Conversely, trained astronauts have in every case, performed assigned tasks. There have been two precautionary delays of scheduled EVAs[2]. It isn't necessarily easy ... there is a lack of initiative, but tasks trained for and scheduled are done and done well (42:A4).

However, Thornton, *et al.* also report that individuals with less extensive training prior to the flight have sometimes been unable to complete all assigned tasks (42:A4). An example was the first journalist in space, a Tokyo Broadcasting correspondent who traveled with a Soviet resupply mission to the Mir space station in December 1990. He had participated in 18 months of rigorous training with the cosmonauts, but still his reports back to Earth were filled with comments about his high level of discomfort and lack of normal function due to space motion sickness and other factors (36:1). In addition, Hettinger maintains that any period of time in which crew members are disabled by SMS symptoms, no matter how severe or protracted, is "extremely critical in terms of the risks and expense involved" (14:424). A 1988 study by Ratino, *et al.* found a correlation between increase in reaction time and magnitude of motion sickness symptoms (32:223). Furthermore, activities outside the Shuttle are not scheduled until after the third day of a flight and are limited to immune or adapted individuals because of the dangers of aspirating vomitus inside the closed environment of a space suit (8:1185)(30:28). This is particularly

[1] In a disabled condition.
[2] Extravehicular activities (space walks).

important because emesis in SMS usually comes suddenly, without the prior nausea or other gastrointestinal symptoms that would normally precede it on Earth (42:A2). So, SMS poses not only a nuisance but also a threat to astronauts and their mission.

## *2.2 Why Does SMS Occur?*

Many researchers consider SMS just another form of the terrestrial motion sickness. They also believe the two syndromes have the roots of their existence in the neural mismatch[3] hypothesis first formed by Reason (33:166). This hypothesis explains motion sickness symptoms as the body's natural response when the information transmitted by the eyes, the vestibular system, and other receptors is not the information the person expects to receive, based on past experience (9:474). This memory of sensory inputs is accessed by a comparator center within the central nervous system (CNS), which correlates them with the current experience and then generates a response dependent upon the duration and the intensity of the existing mismatch. Frank motion sickness would be the result of an "overload" in this comparator center.

For astronauts, discordant sensory inputs are quick to arise upon entry into orbit. Their otoliths, the vestibular receptors dependent upon gravity for determining the head's position, are left without any reference at all. Visual and tactile orientation cues also become ambiguous: "up" and "down" are now redefined constantly by the changing visual and tactile experience of a floating astronaut (28:218).

Although it is widely accepted that SMS is a form of motion sickness, there are other theories which also implicate CNS involvement. The nullification of gravity in orbit affects more than sensory inputs. Some hypothesize that the large fluid shift within the bodies of crewmembers in orbit might directly increase cerebrospinal fluid pressure or chemically change its composition, triggering a response from the CNS

[3]Also known as sensory conflict.

(28:219). Also, similar changes in the pressure or composition of the fluid within the vestibular organs could cause a similar result (28:219).

## *2.3 Motion Sickness Treatments*

Recognizing the probable role of the CNS in motion sickness, researchers have developed three main types of treatments: biofeedback, desensitization, and drug therapy. To date, no method has been completely satisfactory and each has specific drawbacks. Biofeedback trains a person to use relaxation techniques to override the autonomic (involuntary) nervous system and control the physiologic symptoms of motion sickness as he or she experiences them. This approach has had positive effects for some airsick flight trainees but can be very lengthy and requires the person's complete cooperation and concentration (25:106)(18:1153,1157) . To desensitize or protectively adapt someone to SMS, they are subjected to graded levels of the visual and acceleration stimuli believed to induce SMS. The Space Science Board of the National Research Council reported in 1987 that preadaptation had not been effective against SMS, although it did improve the effectiveness of biofeedback training (25:106).

Researchers have also recognized that motion sickness symptoms can sometimes be controlled by drugs. Astronauts and cosmonauts have taken several different drugs, most of which are used to treat terrestrial motion sickness, to attempt to prevent or reduce SMS symptoms. These drugs include scopolamine, an anticholinergic agent[4] which reduces certain inter-neuron communications within the autonomic nervous system, and meclizine, an antihistamine which has inherent anticholinergic effects (25:106)(30:29). However, each of the drugs has undesirable side effects varying from sedation to blurred vision. These side effects pose their own operational hazards which lead researchers to combine drugs in an attempt to offset

[4]Class of drugs which oppose the transmission of acetycholine, a chemical which carries nerve impulses across intercellular gaps.

these undesired effects. A typical example is scopdex, a mixture of scopolamine and amphetamine (30:29). No drug therapy has yet been found which shuts down SMS symptoms and, unfortunately, astronauts continue to experience episodes of emesis while on these medications (19:2).

In an effort to locate a more effective means of preventing or treating SMS, AFIT in 1982 first developed a means of measuring the physiologic symptoms of motion sickness as they progressed. One of the measurements used was the electroen ·phalogram (FEG), which detects and displays the electrical activity of the brain. Despite some earlier discounting of EEG changes during motion sickness (33:63), the AFIT research team found a consistent pattern of high-power, very-low-frequency electrical activity which grew, waxing and waning, as symptoms developed (5:1022,1024)(44:39-40,43-44). The pattern is shown in Figure 1.

The activity was located in a band of frequencies lying below 4 hertz which is commonly called the *delta* and *sub-delta* frequencies (6:111). The research team recognized that although the subjects' external symptoms were those common to motion sickness, their brain activity patterns resembled that of a partial (focal) epileptic-type seizure. (5:1024)

Because the pattern was so distinctive, AFIT began testing the ability of the classic anticonvulsant drug phenytoin to suppress the activity and perhaps reduce the subject's motion sickness symptoms. They discovered that phenytoin therapy provided a mean symptom-free time that was 11.9 times longer[5] than that experienced under the placebo (5:10?4). In addition, none of the experiment participants experienced any side effects other than slight "light-headedness" or a sensation of enhanced alertness (5:1023). Phenytoin treatment also delayed the appearance of the characteristic EEG pattern in most of the subjects, with one of the participant's EEGs never registering the onset of the powerful delta pattern (5:1024). While

[5]With a standard deviation of 6.2.

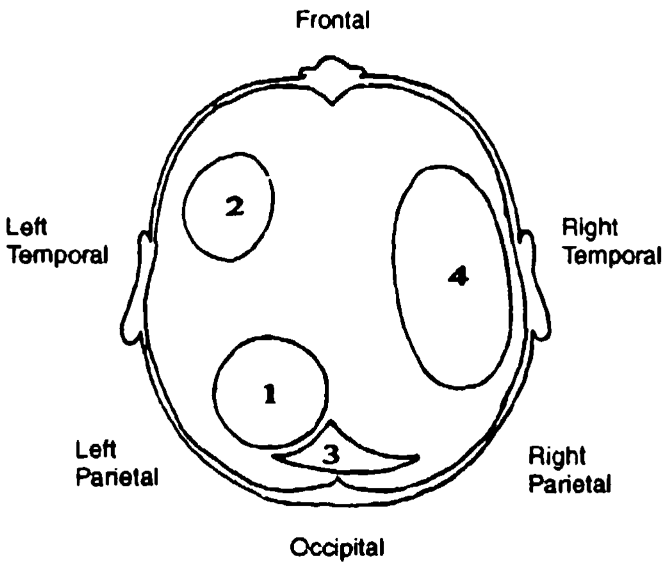


Figure 1. Brain Activity Pattern During Motion Sickness. Vogen reported this signal propagation pattern which increased in signal power with time: 1) Initial symptoms brought a left parietal focus; 2) Almost simultaneously, or shortly thereafter, there was an ipsilateral spread (on the same side) to the fronto-temporal region; 3) A contralateral spread (to the opposite side) follows; 4) More pronounced symptoms brought a right-temporal focus, sometimes unlocalized. (44:48)

the results of the phenytoin trials were very encouraging, they only emphasized the question of how motion sickness could be related to the energetic delta-waves.

To further study the pattern of delta-wave propagation, AFIT has now turned to a different form of brain topographic mapping, the toposcope. The following chapter introduces the field of brain mapping and three major mapping techniques, as well as explains the choice of the toposcope for this next phase of AFIT motion sickness research.

## III. Brain Mapping: A Summary of Current Knowledge

The field of brain topographic mapping, also called brain mapping, exists in order to expand the knowledge of how the human brain functions—and malfunctions. It does so by providing information about the location of brain electrical activity and the spatial and temporal relationships between separate locations. Thus, brain topographic mapping is true to its name: both the prefix *topo* and the word *map* convey the idea of "location." More importantly, brain mapping can offer a significant advance over the usual methods used to describe certain brain phenomena.

Brain mapping is a term describing a group of display methods for which EEG signals are often used as input. The EEG is generated by brain cells as they alter their states of electrical charge by communicating with each other via signals of constantly shifting amplitude and frequency (41:1). These electrical changes extend to the scalp, where electrodes can detect the very minute voltages they create. As these voltages, or potentials, continuously change, they can be traced onto strips of paper to form extremely complex graphs similar to a polygraph (27:168) (see Figure 2). In addition, the "brain print" of these constantly changing signals is different in each individual, as distinctive as the person's signature (46:55).

These rapidly changing lines hold information about more than the surface of the brain. The EEG is the result of both spontaneous changes in brain state and function, and the elicited changes resulting from a stimulus, such as hearing a noise (21:309)(6:1). The tracings of an EEG can even be used to infer the location of signal sources in a third dimension, inside the volume that the brain occupies (26:704). Yet, interpreting the graphs of an EEG with the human eye is often considered to be more of an art than a science, for the EEG contains too much data in an unsuitable form for visual analysis (48:1) and, in all likelihood, a great deal of noise. This is especially true in identifying brain-related diseases and conditions caused by subtle alterations of background activity, such as mental retardation (10:455) or motion sickness. It was

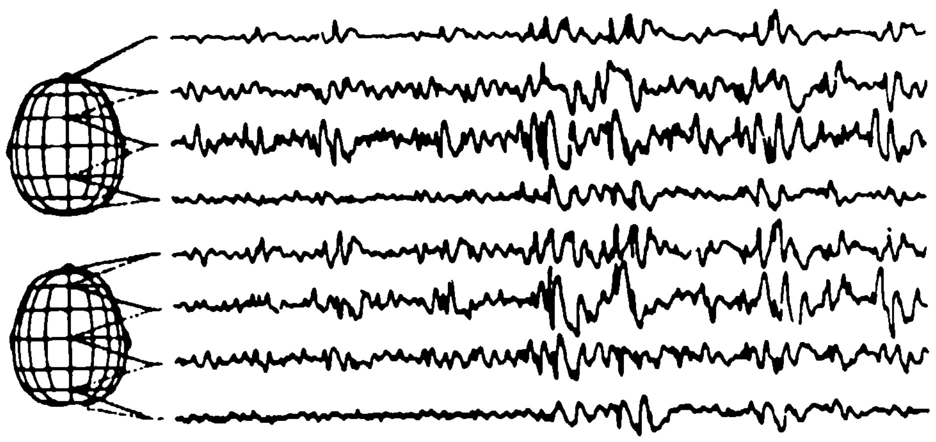


Figure 2. Example of Eight Channels of EEG Tracings (35:73).

because of this data presentation problem that researchers developed the techniques of brain topographic mapping.

Brain mapping is an ever broadening field as new technologies become available. Upon encountering the words "brain mapping," the techniques that might first come to mind are those that produce three-dimensional images: computed tomography, nuclear magnetic resonance imaging, and positron emission tomography.[1] However, these technologies are considered brain-*imaging* methods, as opposed to brain-*mapping*. The field of brain topographic mapping is limited to those techniques applied to data measured using the electroencephalogram (EEG) or the magnetoencephalogram (MEG). This chapter will discuss only topographic mapping methods, and specifically only those applied to interpreting spontaneous EEGs, which AFIT used to record all of its research data. Although brain mapping techniques are also used to study short bursts of brain activity which are induced by specific visual, auditory, or sensory stimuli[2] (16:52), these applications will not be addressed.

[1]These methods are probably better known by their respective acronyms: CT-scan, NMR or MRI, and PET-scan.

[2]This method of recording is known as evoked-potential, or event-related potential recordings.

This chapter will overview several areas. It will begin with an explanation of the role of brain mapping as a research tool. This will be followed by a survey of three groups of brain mapping techniques which appear in the current literature: the Walter-Shipton class of toposcopes (29:14), contour mapping, and statistical mapping. Each discussion will include a description of the technique and its strengths and limitations. Finally, the conclusion will analyze the various mapping methods with respect to the research problem and the goal of continuing the investigation of brain activity during motion sickness.

## *3.1 The Role of Brain Topographic Mapping*

A mapping pioneer, Dietrich Lehmann, compared brain mapping to using a subway map (48:1). He commented that just as a verbal description of the subway system would be greatly aided by a graphical representation of the various routes, so too would raw EEG data be complemented by a brain map.[3] Topographic maps do not add new information, but are intended to make readily available the spatial data locked inside the EEG tracings by showing the data in a space-oriented form (22:29)(24:548). This is accomplished by regarding the electrodes as distributed in two dimensions over the surface of the skull. The way these signals are distributed over this surface is the key to gaining more information as to their origins (6:108).

The field of brain topographic mapping continues to grow with technology and has found a definite place in brain research. H. Petsche said, "The overwhelming invasion of the EEG by mapping methods demonstrates that scientists dealing with the EEG have become aware of the fact that traditional electroencephalography has neglected an essential aspect, namely location" (29:15). The earliest efforts in the brain mapping area, dating back to the 1940s and early 1950s, have given way to computer-based methods which offer unique characteristics to the search

[3]Lopes da Silva, along with several other experts, stresses the importance of not forsaking the raw EEG as the primary record (24:549).

for more insight into brain functions. However, the first device to be discussed, the Toposcope,[4] is especially unique due to its foundational role in the field, the perspective it offers, and its relatively recent reappearance in the literature.

*3.1.1 The Toposcope.* The first brain mapping device used exclusively on humans was built by Grey Walter and Harold Shipton in the late 1940s and called the Toposcope (45:283). Its purpose was to study the intrinsic rhythms, or frequencies, of the brain's internal communications created by the often rhythmic waveforms of electrical, intercellular activity (37:659). Walter's theory was that these rhythms were produced by groups of millions of neurons firing in harmony and could be tied to brain function. In addition, if properly displayed, they would show interdependence between separate areas of the brain as messages were propagated from spatially separated groups of neurons (46:62)(29:14). He shared the vision of Sir Charles Sherrington that the brain was "'an enchanted loom where millions of flashing shuttles weave a dissolving pattern, always a meaningful pattern though never an abiding one'" (47:14). Moreover, he hoped to design the toposcopic display to eliminate the interference of signals in which he had no interest, while highlighting the parts of the brain where activity was related in frequency, as if distinguishing "from the gossip backchat of bystanders and the welter of routine traffic" (46:62).

*3.1.1.1 How the Toposcope Works.* The Toposcope was built in several configurations, but the third model was implemented using the then current vacuum tube technology with 22 miniature cathode ray tubes arranged in the same pattern as the scalp electrodes (39:219) to display information tying together the amplitude and frequency of the brain's activity at each electrode's location. Within each display tube, a vector would rotate at a controlled rate like the hand of a clock about the

[4] "Toposcope" will be capitalized in this report only when referring to a device built by Walter and/or Shipton. The lower case usage will refer to similar tools created by others or to the class of devices.

tube's center (29:14), looking much like a World War II radar display. A photograph of the display, taken in the 1950s, appears in Figure 3.

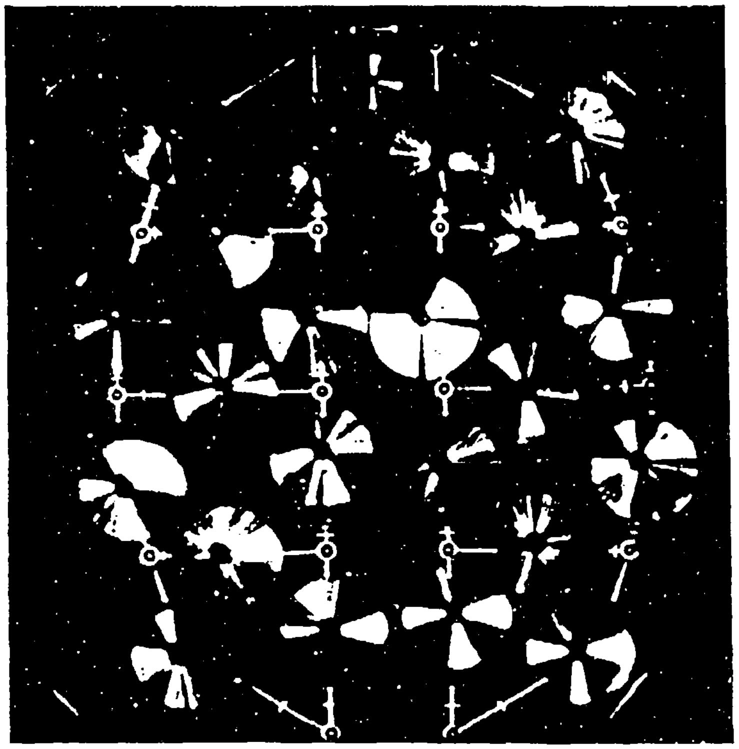


Figure 3. The Toposcope Display of Walter and Shipton. The faint lines between the tubes indicate the linkage of each tube to the electrodes on the head of the subject. The electrode locations appear as small circles between the displays (46:63).

The brightness of the luminous point located at the edge of the sweeping vector was controlled by the signal's amplitude, making the display's brightness instantaneously responsive to the amplitude of electrical signal present at that location by turning off or on the electron beam tracing the display (46:62). The vector rotation rate was manually adjusted, tuning the display to a particular frequency component of the EEG (46:62).

The vectors on all 22 screens rotated in lock-step, but the display regions of light and dark in each screen varied because brightness was controlled in each by a different EEG channel. These displays could not only differ from each other, indicating different dominant frequencies at spatially separate brain locations, but each would also change rapidly over time, indicating the constantly changing frequency and amplitude modulation of the EEG signals. As quoted by Walter and Shipton, the cumulative, mesmerizing effect was summarized by Sir Charles Sherrington as "'...a sparkling field of rhythmic flashing points with trains of travelling sparks hurrying hither and thither'" (45:282).

From this flashing field, Walter and Shipton sought to display relationships between EEG channels. The key to these relationships was the vector sweep rate in the displays. Because the vector in each screen rotated at the same rate, the displays were to present continuous snapshots of the relationship between the 22 EEG channels and the reference frequency set by the sweep rate.

*3.1.1.2 Strengths and Limitations.* The creators of the Toposcope found that when the sweep rate was set to match the frequency of the dominant EEG rhythm, one or more sections of the tubes would become constantly bright whenever they reflected some harmonic of that ever-changing rhythm (29:14). Examples of the wedge, hour-glass, and semicircular patterns that resulted are shown in Figure 4 under the column "Rotational Scan". Phase relations could also be easily recognized by comparing the relative positions of these areas of brightness on different tubes (12:54). Signals of identical frequency, but different phase, presented similar patterns but rotated with respect to each other.

For the first time, the Toposcope made information about spatial and temporal variations in the EEG instantly and simultaneously available (39:217). Screens displaying similar stationary patterns indicated similar frequency signals at those locations. That signal could then travel to neighboring regions, causing their screens

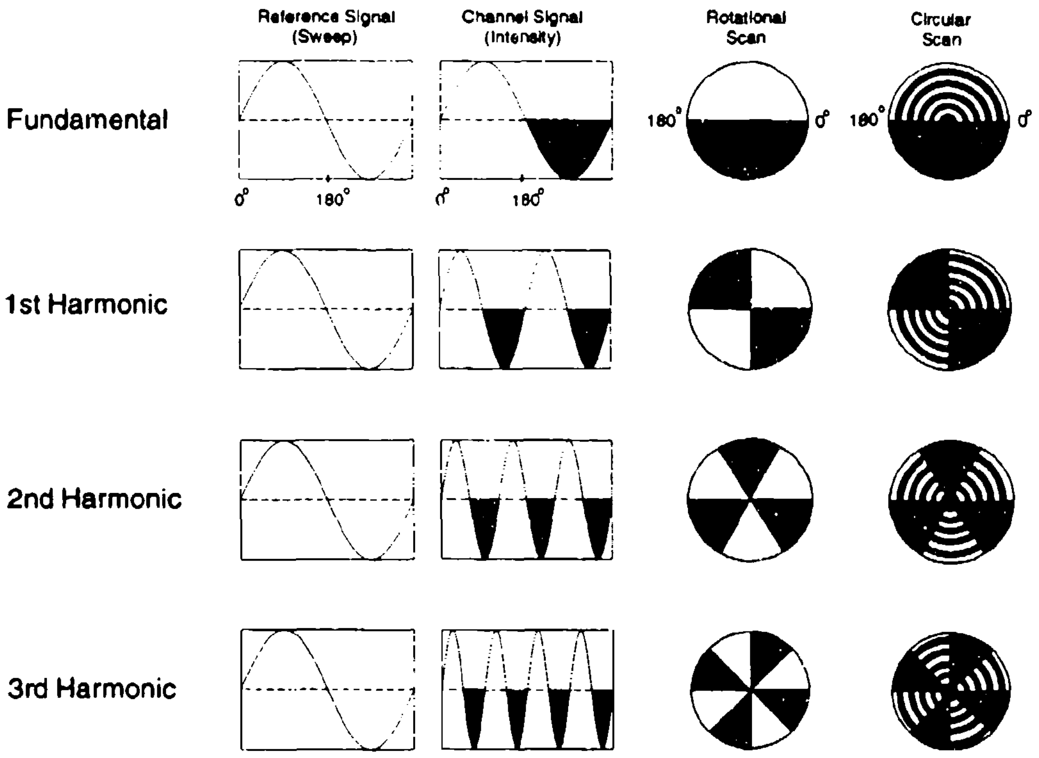


Figure 4. Examples of Toposcope Displays of Simple Signal Relationships. In each, the 1-Hz reference signal is used to correspond to one sweep of the rotating vector. Positive amplitudes of channel signals create bright regions on the screen at locations corresponding to their position in time. If these frequencies were constant, they would result in stationary patterns of the same design.

to show the characteristic pattern. In addition, a signal could be considered to originate between two adjacent locations whose screens displayed the same frequency pattern, but 90 degrees out of phase (45:289). However, a consistently phase-shifted, but similar, stationary pattern occurring between two separate areas of the brain could result from a propagation delay of the same signal and point to an origin lying other than between the two electrodes. The Toposcope highlighted those frequencies harmonically related to the sweep frequency and filtered out others because related frequencies created stationary patterns and unrelated ones created patterns that drifted around the screen, never remaining in one place long enough to "summate" for the eyes or film (38:484)(45:287).

However, the Walter-Shipton toposcope did have some limitations.

- It was difficult to interpret. Walter wrote, "When we began to use this machine, we found the time maps hard to understand. But gradually the new code has begun to penetrate our thick heads and much of what was quite bewildering in the ordinary brain prints now seems to be taking on new form and luster" (46:62).
- It displayed only the signals at the electrode locations since it performed no interpolation between electrodes, thus limiting its spatial resolution (39:220).
- It was incapable of exact amplitude measurements, but such measurements were not among its intended purposes (45:285, 287) (39:218, 220). Shipton considered this a necessary sacrifice of any area-display seeking spatial and temporal resolution (38:485).
- It required photography as an integrating mechanism over several revolutions to extract and capture the stationary patterns (46:62) (45:289).
- It was difficult to set the speed of the servomechanism controlling vector rotation because of the operator's tendency to focus on moving rather than stationary patterns and because the frequency of the EEG is not stable (38:484).

*3.1.1.3 The Updated Version of the Toposcope.* Additional versions of toposcopes were created later by Shipton, with the most recent being announced in 1981. In building this latest toposcope, Shipton updated it with a mixture of analog and digital technology, but retained the concept and type of display and features from an earlier, interim toposcope announced in 1963 (37:659)(38:483). This display, christened the "circular scan,"[5] altered the original rotational sweep to cause it to inscribe a spiral on the screen by using a luminous point positioned at the end of a rotating vector of increasing length. This added a fixed-length time axis to the display, making it possible for an observer to see a brief history of the most recent signals for comparison. Figure 4 provides examples of this display under the column "Circular Scan." Signals harmonically related to the rotational speed appeared on the new display at the same angular position but at an increased radius on each successive sweep (38:484). Those signals differing slightly from a harmonic relationship would result in a drifting pattern now more easily recognized. Figure 5 illustrates these features as it compares the temporal and frequency resolution of the initial and revised displays.

Now time could be read over short periods by measuring the distance of any portion of the spiral to the screen's center and photography was only necessary as a recording medium and not an integration mechanism (38:484). In addition, the circular scan system could more easily differentiate between signals of slightly differing frequency (39:220). The time period of the display was a maximum of 60 sweeps (38:485).

An additional feature of the new Toposcope was the ability to use the signal in a selected channel as the reference so that the activity of the entire brain could be correlated and compared with it (38:484). By measuring the frequency of the reference signal from the intervals between time-axis crossings, that signal would drive the rate of vector rotation in the displays of all channels (38:490), thus eliminating

[5]Also known as the "helical" scan (39:14).

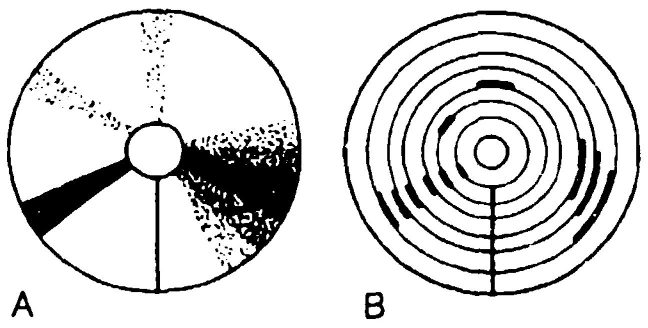


Figure 5. Comparison of Resolution of Original and Updated Walter-Shipton Displays. *A*. The photographically integrated original display is ambiguous when presented with short duration signals or those slightly offset from the frequency of interest. *B*. The circular display clarifies these ambiguities, showing brief signals in the 10 and 12 o'clock directions, a steady, harmonic signal at 8 o'clock, and a steady, but non-harmonic signal at 4 o'clock, resulting in a constant shift in subsequent traces (39:220).

the need to manually control the rotational frequency. The rotational speed could be related either to the fundamental frequency or to some sub-multiple of the signal by use of an attenuator (38:490). Although Walter did mention that the sweep rate on the original Toposcope could be controlled by signals from the subject's brain (46:62), no indication of how this was done could be found in the literature.

This 1981 version of the Toposcope made several positive contributions but had its limitations as well. By virtue of the new circular display, it did remove the need for photographic integration in order to focus on stationary patterns in the display. The addition of an automatic mode which drove the sweep rotation according to one EEG channel eliminated the restrictions and difficulties of manual control. Also, Shipton reported that this system was better suited than the original Toposcope for studying spontaneous EEG, as opposed to evoked potentials (38:485). However, its limitations included:

- It did not address the limitations of its predecessor in spatial resolution and amplitude measurement accuracy.
- It was unable to display evoked responses to slow, repetitive stimuli (rates below 3 per second) without losing adequate frequency resolution (38:485).

The Toposcope continues to offer a unique perspective on brain activity research, displaying instantaneous relationships in frequency and phase between separate areas of the human brain (39:221–222). But its Shipton-defined role as "an extremely sensitive *indicating* device, rather than a precise *measuring* instrument" is an important distinction (38:485).

*3.1.2 Contour Brain Mapping.* Ever since Antoine Rém^nd created the first chronotopogram in the early 1960s (35:73–93), contour mapping has been an efficient and popular way of overcoming the problem of discrete EEG sampling in space (23:145). Contour mapping, unlike the Walter-Shipton class of toposcopes, uses interpolation methods in order to smooth out the informational discontinuities between electrodes (23:143). The output is an image very similar to a contour map of a geographic area, with free-form curves indicating areas of similar elevation and the spacing between curves showing an incremental elevation change.

*3.1.2.1 How Contour Mapping Works.* There are two basic groupings of contour mapping techniques (26:705). One consists of recording scalp potentials from a two-dimensional array of EEG electrodes at a single moment in time and creating a map of the potential field using interpolaticn methods. Thus a series of maps can be produced showing changes over time. The second group records only from a line of electrodes which measure the EEG potential at specific points along one dimension. However, the recording is continuous, producing a two-dimensional plot with the time axis running the length of the output. Here, too, an interpolation scheme is used to estimate those points not measured. The chronotopogram belongs to this class of contour mapping (26:705). An example appears in Figure 6.

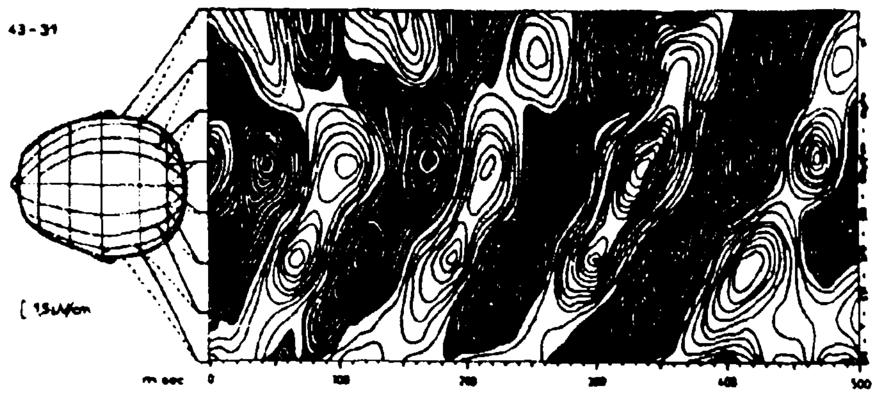


Figure 6. Rémond's Chronotopogram. An example of the first contour brain map (35:81).

There are several different means of interpolation in use, both linear and non-linear. Although interpolation is a common practice, its use should not be considered trivial. Instead, careful consideration should be given to selecting the best interpolation method since 99 percent of the resulting brain map will result from the algorithm's calculations, rather than from the measured, known points (27:168). The most common linear methods used in brain mapping applications are the *nearest-three-neighbor* and the *nearest-four-neighbor* algorithms, according to Wong (48:14). They cause the interpolated values to reflect the trend set by the values of the bounding three or four points. Variations of these linear methods differentially weight the bounding or spatially distant points to affect the calculated value (11:21). In non-linear interpolation, the calculation fits a complex curve to the data points and uses the resulting equation to determine the needed values.

Sometimes, averaging procedures are also used. Measurements of potentials at each electrode are averaged together over some predetermined period of time and then these average values are mapped as before (26:705). This results in the loss of information on high frequency changes in brain activity, but works to highlight the most significant activity patterns since they were present during all or most of the averaged period (48:12). A secondary benefit is the resulting data reduction in generating only a single average map (48:12).

Contour maps are not limited to displaying instantaneous voltages, but can also display voltages from portions of a signal's frequency spectrum. The EEG data can be processed by a computer-based mathematical technique called a fast Fouri· · transform (FFT) which transforms a signal stream, often 1-to-10-seconds long, from the time domain into the frequency domain (13:348). The resulting individual frequency components could then be summed back together to reproduce the original signal. These components can be displayed for a single electrode as traces in which the vertical axis is voltage or power and the horizontal axis is frequency. However, the spectrum region for the entire array of electrodes can also be displayed as a contour map, affording an image of the amplitude of overall brain activity in that frequency band (48:24).

Artificially coloring the regions of a contour map can highlight the areas of similarity and dissimilarity, and therefore is a very popular option (27:168). Shades of gray or variations in the density of dots or other symbols are comparable highlighting methods.

*3.1.2.2 Use of Contour Brain Maps at AFIT.* Both Banducci and Vogen used contour maps in analyzing motion sickness EEG patterns (2:86)(44:20-21). Both researchers used the contour maps created by a software package called the Brain Atlas[6] to display voltage maps from the delta region of the brain's frequency spectrum. Examples of these color maps are reproduced in black and white in Figure 7.

[6]A product of Bio-logic Systems Corporation.

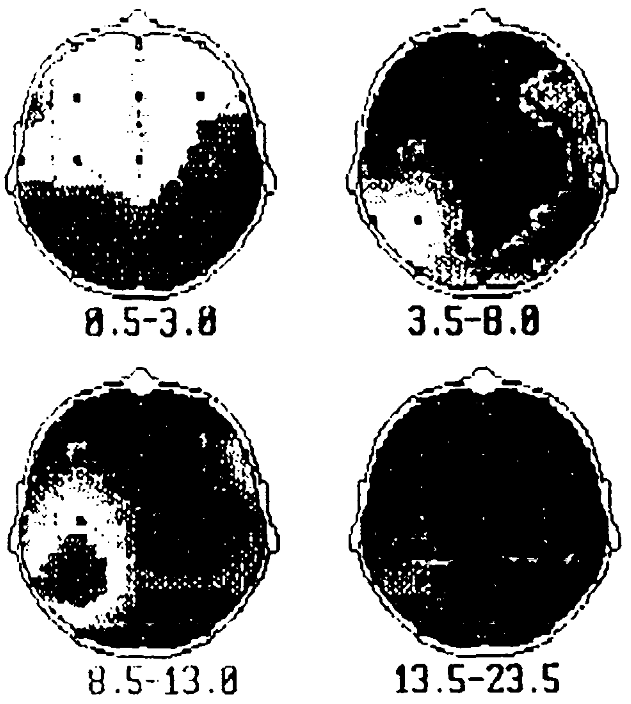


Figure 7. Examples of Contour Brain Maps Used in Prior Research at AFIT. These four maps display activity in four different frequency bands, as labeled by the numerical range below each map.

They processed 14-channel EEG data using FFTs to transform 2-to-10-second epochs (blocks) of data into the frequency domain (44:32). The resulting maps were produced in color and clearly displayed the spread of an electrical activity pattern as motion sickness symptoms grew in severity.

*3.1.2.3 Strengths and Limitations.* Contour mapping offers two distinct advantages to the field of brain mapping. First, it presents data in a manner easily assimilated, even by people who know relatively little about EEGs (27:168). Second, when used in conjunction with the FFT, contour mapping can present this data in either the time or frequency domain, permitting researchers to look at brain activity across all frequencies or by frequency region.

Despite the power of the FFT function and the appeal of the smooth, colorful display, there are several acknowledged drawbacks to the contour mapping approach. The following are precautions to consider when interpreting a contour brain map.

- The vast majority of the mapped values are interpolated, and the type of interpolation determines how the peak voltages should be considered. In linear methods, any peak voltages are forced to lie at the locations of the electrodes rather than at a location of a calculated value (48:14). In non-linear interpolation, voltage peaks can occur at any location, but this method can also assign maxima where none actually exist (48:16). Thus, it is important to remember that all interpolated points are *approximations* (11:21).
- High frequency changes in brain activity will not appear if average values were used in constructing the display.
- Fast Fourier Transform maps display integrated, not instantaneous, values (7:342). Specifically, the frequency band power will correspond to the standard deviation of the average of all momentary maps over the time period (20:58-59). Shipton also states that additional difficulties and uncertainty are

introduced in the low frequency regions when attempting to splice together a reliable picture of the average activity over time (39:218-219).

- Both the number of electrodes and the reference voltage used can drastically affect the appearance of the contour map (11:23)(26:705).
- It is possible to misuse color in the contour map. If contrasting colors are used to represent neighboring amplitude ranges, they can lead to the interpretation that an important threshold exists, when it really does not (11:23)(39:223).

These limitations of contour mapping apply to the previous research at AFIT based on brain mapping. While Vogen was specifically careful in selecting the time periods (epochs) he used in averaging his samples (44:32-33), the mapping method used in his and Banducci's efforts could not display the frequency-dependent, *momentary* changes in the EEG pattern, but only averaged changes which were "blurred" in the time domain. In addition, the majority of the hundreds of displayed pixels in any given map were generated by interpolating only 14 data points. In order to produce a time-sequenced approximation of the propagating pattern, Vogen produced an animation using many separate maps averaged over overlapping periods of time recorded on a video camera (44:34).

*3.1.3 Statistical Brain Mapping.* In statistical mapping, the image created does not represent the voltage or power of brain activity. Instead, it maps statistics comparing each point of the EEG spatial field with some reference, with the goal of indicating the degree of difference among images. The introduction of statistical techniques has greatly changed the outlook for brain topographic methods (39:222). Gevins further emphasized the role of this approach:

> Modern information processing of EEG recordings brings 3 essential advantages: 'precision (and accuracy) of measurement,' 'speed of processing,' and 'significance,' that is, a new understanding of the results with

the help of statistical tools. Any statement which possesses great significance must derive from a large enough number of accurate measurements and be made after a statistical operation accompanied by a strong enough statistical 'weight.' (34:viii)

*3.1.3.1 How Statistical Mapping Works.* Statistical mapping usually begins with a plot of the brain potential field in some or all frequencies produced by a contour mapping method. It then performs a pixel-by-pixel statistical comparison with a reference map which can be another individual image or, more commonly, a map of mean values from some population. The results are then transformed into a new image via a statistical to optical transformation (10:455)

In the case of significance probability mapping (SPM), Duffy, *et al.* use two different type of statistics: the Student's *t*-test and the *z*-transform to delineate regional topographic differences in brain activity (10:456–457). In the *t*-statistic form of SPM, the individual points are summed between multiple maps within two separate sets: the control and experimental sets. The results are two maps consisting of the summed pixel values of their respective sets. Each of these images are then converted to a map of the mean values and a map of the variances. Finally the *t*-test is applied between the control and experimental sets and the result is the *t*-statistic map, which reveals the regions in which the differences between the two populations are statistically significant (10:457). In *z*-statistic SPM, an individual map is compared against the mean and variance maps of reference population. The *z*-transform map results, where the values are the number of standard deviations the individual map lies from the mean of the reference group (10:457).

Another approach to statistical mapping is the neurometrics method developed by E.R. John, *et al.* which estimates the probability that the quantitative measurements of brain activity reflect dysfunction (31:153). Neurometrics first extracts particular features from the EEG data. This method then produces brain maps using the *z*-transform with the scale representing the probability that the values shown might be found in a healthy, normally functioning person of the same age.

John has also demonstrated the capability of neurometrics to accurately classify the transformed individual maps of psychiatric cases according to the disorder present (17:116).

*3.1.3.2 Strengths and Limitations.* As Rémond's quote pointed out earlier, the strength of the statistical methods lies in their power to highlight the significant differences in brain activity. Another advantage is the capability to then dependably classify maps into categories of dysfunction. All this increases the usefulness of brain mapping as a research tool.

The limitations of the method are:

- "Normal" is a relative label. There is significant debate over defining the statistical distribution of the normal EEG. Different techniques sometimes use substantially dissimilar limits of normal (27:171).
- Medications, age, gender, and handedness[7] affect the statistical comparison. These factors must be considered because they affect the EEG and could invalidate any comparison to a normal population (27:171).
- Abnormal results are not necessarily significant. A computer can easily generate 5000 or more statistical results. Thus, a few results outside even a three standard deviation criterion for abnormality could be expected due to random chance (27:171).
- Statistical mapping requires a sizable normative database of maps, a significant investment of resources (11:25). Location-by-location comparisons between individual maps cannot provide information about the significance of any differences because no variances are available. Instead, individual maps should be compared only against reference populations (21:345).

[7] Whether the person is right- or left-handed.

*3.2 Further Application of Brain Mapping at AFIT*

Each group of brain mapping methods has distinctive strengths and limitations. Because of this, they would seem to complement each other: a toposcope displays instantaneous timing and phase relationships, contour mapping "fills in the gaps" between electrodes and provides spectral maps, and statistical mapping calculates the significance of differences between maps.

However, with respect to AFIT's quest to learn more about brain activity patterns during motion sickness, each technique may or may not be practical. Contour mapping of FFT-generated spectrums was the method used in the initial discovery of the pattern of interest. Further application of this technique does not appear to offer the most hope of additional insights. Statistical mapping, while a very powerful tool, requires a sizeable reference database which AFIT neither has nor will be able to afford in the near future. A toposcope like that of Walter and Shipton will reveal more of the brain activity pattern which motion sickness "weaves." It is especially suited to investigate instantaneous spatio-temporal images and should be tunable to the delta/sub-delta frequency region. Although less complex in implementation than contour mapping with its interpolation and FFT operations, the toposcopic technique will nevertheless offer a continuous, alternative display of brain activity patterns as they spread over the brain's surface moment by moment.

Both Walter and Shipton saw the toposcope as more of a qualitative rather than a highly quantitative tool, stating that encephalographers are usually more concerned with the frequency and spatial location of signals than their exact amplitude (38:483). Shipton wrote in 1986,

> Numbers are not (Lord Kelvin notwithstanding) the only route to knowledge. An analogy can be found in the analysis of handwriting. It is exceedingly difficult to represent a signature by a polynomial; a bank clerk can quickly recognize a valid or invalid signature even in the presence of considerable artifact (39:222).

Given the dual qualitative/quantitative nature of AFIT's search for motion sickness brain activity patterns, the desire to use a different topographic method to compare against the previous applications of contour mapping, and the lack of the sizeable database required to implement statistical mapping, the toposcope was chosen as the tool for this latest phase of motion sickness research. The AFIT toposcope will function similarly to that of Walter and Shipton. Furthermore, while the limitations of spatial and amplitude resolution are characteristic of the toposcope class of devices and will not be improved upon by AFIT's new toposcope, the use of computer code instead of electronics as the display's building blocks offers new flexibilities, features, and interpretation aids. How this tool was implemented is described in the next chapter.

# *IV. Research Approach*

There were three major efforts which comprised this research. The first was to access and convert the binary EEG data in the Bio-logic computer files in the Motion Sickness Lab. The next task was to design and construct a toposcope by writing software using C to drive a computer graphics terminal. The last task was to run test signals and digitized EEG data through the toposcope, making adjustments to the display and frequency selection, in order to characterize the toposcopes performance and study the brain activity pattern in the delta frequency band. This chapter will describe the approach used in each of these areas as well as list the required material and equipment

## *4.1 Preprocessing the Data*

The first task was to ensure digitized EEG records would be available to drive the computer-based toposcope. No additional recordings could be made due to the loss of funding for AFIT motion sickness research at the end of FY91. Records from 1991 were available on 14-channel beta tapes, with the data recorded in analog form. Each of these records had also been previously digitized using the Brain Atlas software and stored on diskettes or a hard disk, but the format of the data within each file was unknown.

The data preparation task is documented in Appendix A and consisted of the following steps:

- STEP 1.1 - Determine format of data in Bio-logic files.
- STEP 1.2 - Validate file interpretation and voltage values obtained.
- STEP 1.3 - Move data files to Silicon Graphics machines and convert to UNIX format.

- STEP 1.4 - Read pertinent file header data, reorganize data by channels, and read in data points.
- STEP 1.5 - Validate output.

## *4.2 Designing and Coding the Display*

The next task was to design and construct a toposcope on a computer graphics terminal. The steps in this process follow:

- STEP 2.1 - Learn basic C syntax and grammar, as well as the fundamentals of 4Sight, the Silicon Graphics windowing system.
- STEP 2.2 - Obtain public-domain graphics code as similar as possible to the desired product and learn how it functions.
- STEP 2.3 - Construct display prototype program from sections of original and borrowed code.
- STEP 2.4 - Build test data files using known signals (sinusoids in the delta frequency range).
- STEP 2.5 - Test Phase I display using test data files. Test display's ability to highlight harmonics of sweep frequency to a human observer.
- STEP 2.6 - Construct Phase II toposcope based on lessons learned.

Instead of the circular displays of the Walter-Shipton toposcope, square displays for each channel were chosen for the AFIT version because of the relative ease of creating and updating moving lines on the Silicon Graphics workstations. The display for a given channel consists of a horizontal, gray-scale "bar" of fixed dimensions which moves up and down within the region of a box. The boxes appear on the display in the same relative orientation as their sources on the subjects' scalps. The intensity of all bars is determined by a single signal or channel, known as the reference, while the vertical displacement of each bar within each of the 14 channel boxes is driven independently by the signal of that channel. This results in the brightest bands appearing in all channels whenever the amplitude of the reference signal is peaking, but at locations varying across channels due to the different instantaneous amplitudes of the EEG in each channel. Figure 8 shows the original concept of the AFIT toposcope while Figure 9 gives simplified examples of expected displays of sinusoidal test signals.

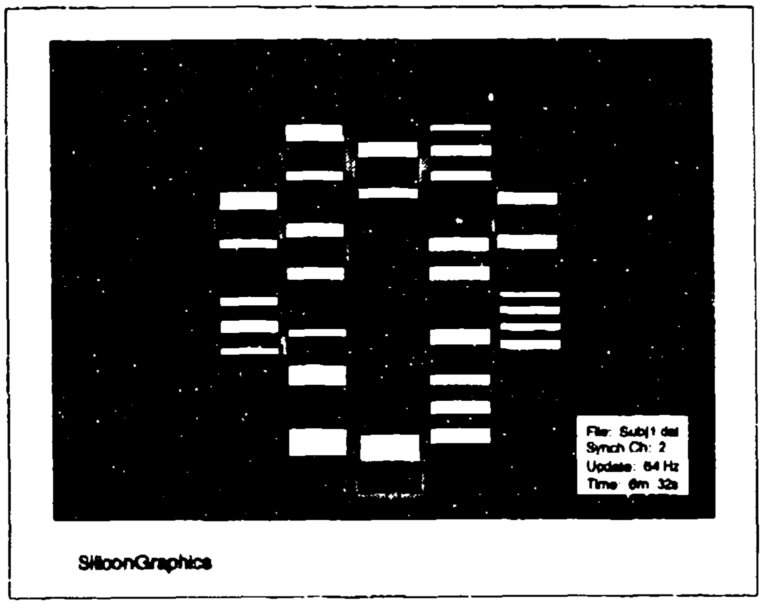


Figure 8. Original Concept of the AFIT Toposcope. Note that horizontal traces replace the circular sweeps of the Walter-Shipton version.

Harmonic relationships between reference and channel signals would be indicated by one or more stationary bands of maximum intensity in that channel's display (see Figure 9). However, it should be noted that because of the difference in the role of the reference signal between the Walter/Shipton and original AFIT toposcopes, different harmonic relationships are displayed. The original toposcope used the reference signal as an angular displacement driver to set the speed of the rotating vector and driving the display to key on *harmonics* of that reference frequency. For example, a stationary hourglass pattern (two opposing wedges) indicated that the

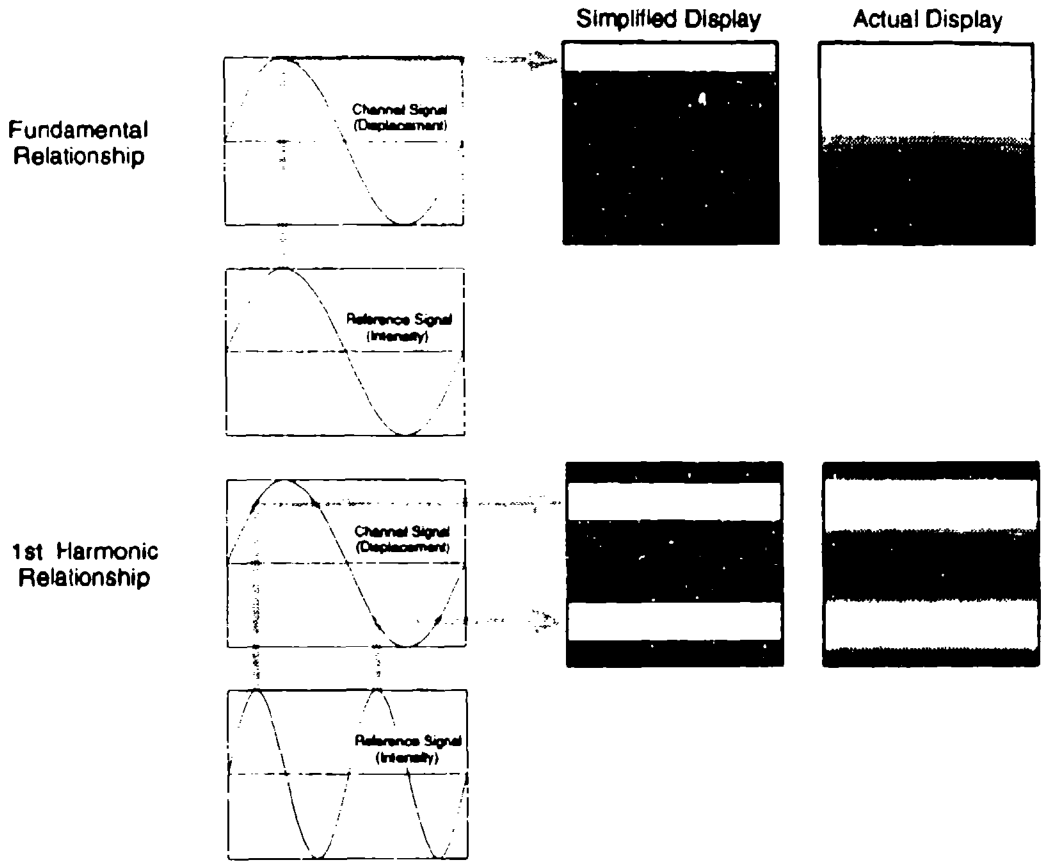

Simplified Display
Actual Display
Fundamental Relationship
Channel Signal (Displacement)
Reference Signal (Intensity)
1st Harmonic Relationship
Channel Signal (Displacement)
Reference Signal (Intensity)

Figure 9. Expected Test Displays on the AFIT Toposcope. Input signals are sinusoids. The reference signal determines the brightness of the bar (only the peak brightness is shown in the simplified version) and the channel signal determines the vertical position of the bar within the box. Note the intensity gradients occurring in the actual display. Displays shown are not instantaneous but represent the composite display as seen by an observer.

channel signal was a second harmonic of the reference signal. But in the original AFIT version, the reference signal drives the brightness of the displayed bar and the channel signal drives the displacement, vertically in this case. If a double-bar stationary pattern results, it indicates that the channel signal is potentially a second *sub-harmonic* of the reference signal.

Thus, while both toposcopes indicate channels whose signals are at the reference's fundamental frequency, they also highlight signals on opposite sides of the reference frequency. This role change for the reference signal was required by the design chosen for the AFIT toposcope and hardware restrictions, all of which are explained in Chapter V. Also, a means of automatic frequency selection was implemented in a manner similar to Shipton's 1963 toposcope, as discussed in Section 3.1.1.3.

## *4.3 Analyzing Test and EEG Signals with the AFIT Toposcope*

The final task was to use the motion-sickness-affected EEG records as inputs to the completed toposcope and analyze the resulting displays for evidence of frequency-based relationships between separate areas of the brain as reported by Vogen. The steps in this process follow:

- STEP 3.1 - Use test signals to determine the toposcope's response to signals sharing fundamental, harmonic, sub-harmonic, and phase-shifted relationships. Adjust the display as necessary based on lessons learned.
- STEP 3.2 - Determine from Vogen's notes the time-spans in the EEG record where the subjects began reporting motion sickness symptoms and in which channel(s) the high-power, low-frequency EEG pattern characteristically appeared.
- STEP 3.3 - Transfer the EEG data record to the Silicon Graphics workstation where the AFIT toposcope was resident.

- STEP 3.4 - Display the record on the toposcope, referencing the signal of all the channels to a fixed-frequency reference, as well as to EEG channels determined to be active in the pattern.

The expected result was that the display of the reference channel would initially show a stationary, single bar while other channels showed no meaningful pattern. However, as the high energy, low frequency signal began to propagate across the brain, the corresponding patterns would fade into and out of the reference channel pattern, perhaps harmonically related and/or phase-shifted, as the activity pattern waxed, waned, and spread. The pattern was expected to follow that described by Vogen and shown earlier in Figure 1, Section 2.3.

## *4.4 Chapter Summary*

This phased approach, accessing the data, constructing the display, and then studying the toposcope's presentations of the data, was intended to provide a systematic means of using a new tool to further study the brain activity patterns during motion sickness. While similar in many ways to its predecessors, the AFIT toposcope uses a unique display methodology. The first step, EEG data record conversion, is discussed in Appendix A. How the new toposcope was designed and coded to display the converted signals is the subject of the next chapter.

# *V. Building the Display*

The program for the graphics display and user interface of the AFIT toposcope grew from a foundation formed by a "host" program *Profile.*[1] Because of the absence of C and graphics programming experience, this method of using another program as a learning tool and foundation was adopted by the researcher. The program chosen was the result of an earlier AFIT computer graphics project to modify the demonstration program *Curve Demo* provided by Silicon Graphics for users as public-domain software. *Profile* was chosen because it used windows, drew lines, ran on the Silicon Graphics workstations, and at least one programmer at AFIT knew how it operated. The display portions were eventually gutted and rebuilt, but the event handling[2] portions were left largely untouched.

## *5.1 From Code to Display*

There were several tasks in this portion of the research: boxes and bars were to be drawn on the screen; the bars were to move in sequence with an incoming signal; and the brightness of the bars was to vary in accordance with a separate signal.

*5.1.1 Drawing to the Screen.* The graphics library functions of the Silicon Graphics IRIS-4D workstation used for the project made the display of the simple objects required relatively straightforward. Functions existed to draw boxes as well as lines of varying widths. The bars used in the display were actually line segments which were 2 to 30 pixels wide. The vertical displacement of the bars resulted from tying the y-coordinates of the line segment's vertices to the value of data in that channel, which was clipped at three standard deviations from zero and then scaled to use the entire height of the box. Initially, the intensity of the bar was controlled by

[1]Written by Capt Chuck Wright.
[2]Refers to means of incorporating operator actions into program execution.

similarly clipping the reference channel's input and scaling it to the 256-value gray scale. Thus the bar would appear very dark at a minima of the intensity driving signal, very white at a maxima, and some shade of gray at points in-between.

*5.1.2 Animation Using Double Buffering.* Smooth animation was possible because of the technique called double buffering. The method involves displaying a front buffer (bit plane) while updating a hidden, back buffer. These buffers are then switched and the buffer which was earlier on display is now updated in the background. This process continues as long as required. Because the buffers can only be switched as fast as a screen can be drawn or refreshed, animation frequency is limited to 72 Hz. Attempting to animate an object on a single bit plane, even a simple object such as a bar, results in a jerky and confusing display. Using this double-buffer technique, a previously drawn object, such as the display bar in a channel box, could be erased from its old position and redrawn at a new position while in the back buffer and then the two buffers switched, displaying the newly drawn screen. When this operation was repeated rapidly it resulted in the bars' smooth motion on the screen.

## *5.2 The Phase I Display*

A prototype display was coded and produced a single window which contained two boxes, corresponding to two EEG channels. Within each of these boxes was a single, horizontal bar whose deflection would be controlled by the frequency of one EEG channel while the brightness of the bar would be determined by the amplitude of a second channel. If the two channels were of equal frequency, a single band of brightness should be stationary in the box. If the signal driving the brightness was a harmonic of the signal driving the deflection, then more than one stationary band would be displayed. In addition, if the signals differed only slightly in frequency, the brightness bands would slowly drift up or down within the box.

The initial display was difficult to interpret because too much information was being displayed. The smoothly, but rapidly changing shade of the bar as it traveled the height of the box was interesting, but confusing, making it hard to track the locations of the brightest bands on the screen. A wider bar (20 vs. 5 pixels) and the decision to only display the bar when the intensity driver was at an amplitude peak helped significantly. Since the number, location, and drift of the brightest bars was the key to interpreting the display, both of these changes worked well in focusing the viewer's attention on the important aspects of the display. However, it was this change which also determined the role of the reference signal in the display. Since the display was only drawn as indicated by the occurrences of peaks in the intensity driver, the display would be faster if all 14 channels were updated with each screen redraw event. This mandated that the reference signal, since it was common to all channels, should be the intensity driver. To do otherwise would significantly slow the display update rate.

With these modifications, the Phase I display provided the displays expected when driven with sinusoidal test signals between 0.5 and 2 Hz. The display successfully identified first and second harmonic relationships between brightness and displacement driving signals. These test signals were generated on a function generator, verified on an oscilloscope, digitized using the Brain Atlas, and converted using the **prep_data()** function in the toposcope code. While test inputs could have been generated by a C program and directly fed into the display, merit was seen in using as many as possible of the same steps to create the test data as were used to preprocess the EEG data files.

## *5.3 The Phase II Display*

While the preceding display had established the basic principles used in the display of the AFIT toposcope, the Phase II toposcope created the tool which was better suited for the EEG data analysis which was to follow. It was during this period that the AFIT toposcope was given the name *TOPOS*, the Greek word for "place." Appendix I contains the code for the TOPOS program with associated comments.

*5.3.1 Appearance Modifications.* The two boxes in the Phase I display were replaced with 14 smaller ones, arranged in the pattern in which EEG electrodes where placed during data collection. Text was added to clearly label the channels and the standard electrode locations for reference purposes. Status and configuration information was written to the screen to keep the user appraised of current data file, elapsed time, reference channel, and selected display options.

*5.3.2 Display and Processing Modifications.* Vertical displacement in the displays was no longer scaled to each channel's maximum and minimum values because extraneous spikes in the data caused the majority of data points to be compressed into a narrow region. Instead, the upper and lower limits of bar travel were set to three standard deviations of each channel, using values calculated during data file conversion.

A significant change was the introduction of a time axis in the display, similar in principle to Shipton's circular display. The display of TOPOS changed so rapidly that the advantage which Shipton found in his circular scan over the original rotational scan became very apparent; TOPOS also needed a means of visually tracking the display's history. The time axis of the circular scan was adapted to TOPOS in the same linear fashion as its display: time "bins" were created in the software and each call to draw the bar wrote only a vertical slice of the bar as wide as one bin and right-shifted but adjacent to the slice last drawn. An example is shown Figure 10.

Thus a time record of the bar's path in each channel display appeared on the screen, being erased only as the display wrapped around in continuation of its task. All this permitted the observer to clearly note the bar's path over the last 69 draw events at any time during the display's operation. This time-shifted display method

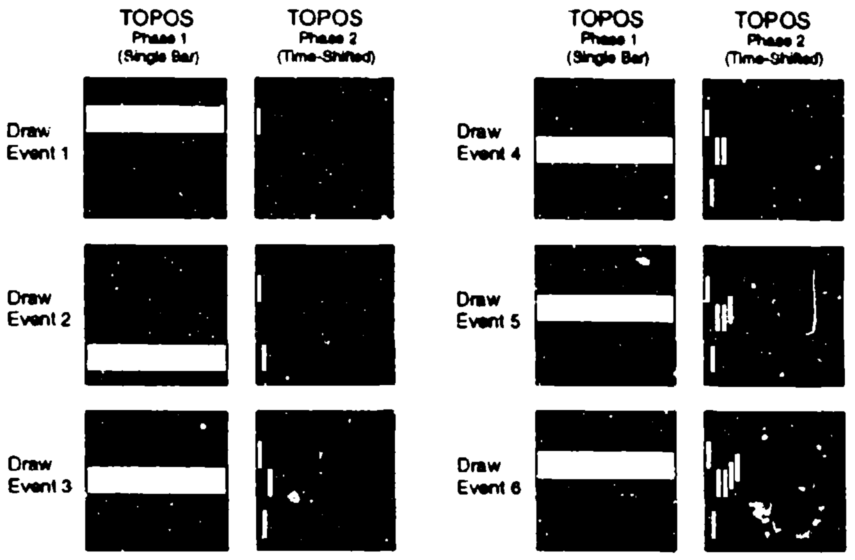


Figure 10. Time-Shifted TOPOS Display. Note the visual history of the display bar's path not provided by the single-bar display.

resulted in the first clear indication of a fourth-harmonic relationship between signals during TOPOS testing.

Five other modifications followed, all targeted at improving the display's ability to focus on lower frequency components of the EEG. The first was the option to trigger draw events at axis-crossings (sign reversals) of the reference signal, rather than at its amplitude peaks. Figure 11 provides an example of signal-display relationships in both the single-bar and time-shifted displays using this event trigger. This option was required by the need to study the lower frequency components of the EEG signals. While peak-driven drawing was satisfactory for the smooth, sinusoidal test signals, the technique keys to the highest frequency components of an EEG signal. This led to the second modification: using "smoothed" EEG inputs. Axis crossings, especially those of an EEG signal that has been smoothed, occurs at the signal's lower frequencies. Smoothing was necessary because energetic, high frequency signal components were often large enough in amplitude to drive the signal

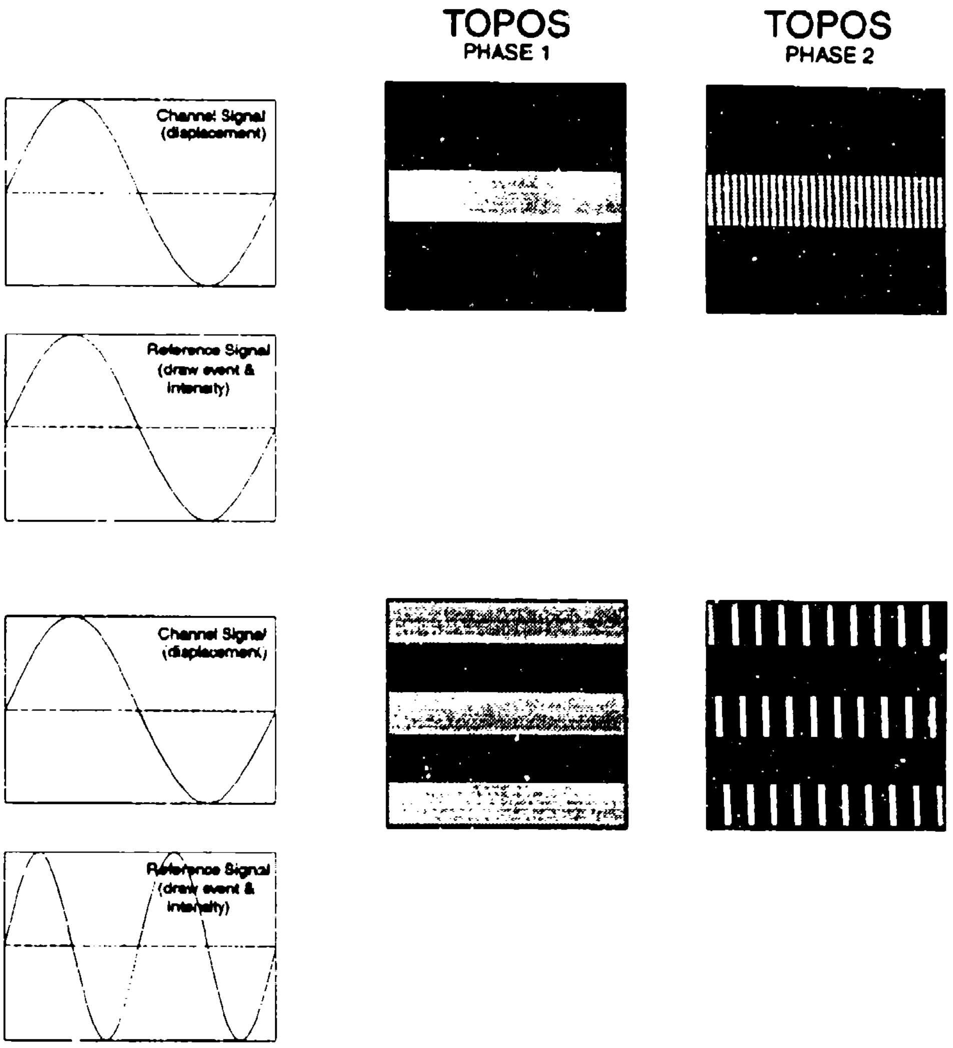

TOPOS
PHASE 1
TOPOS
PHASE 2
Channel Signal (displacement)
Reference Signal (draw event & intensity)
Channel Signal (displacement)
Reference Signal (draw event & intensity)

Figure 11. TOPOS Test Displays Resulting from Reference Signal Axis Crossings. Display intensity is muted in comparison to reference signal peak-generated displays and three bands now indicate a first harmonic relationship.

across the axis while the mean local amplitude of the signal was still well above or below the x-axis. The relationships between the peaks and axis-crossings of an EEG signal and the benefits of signal smoothing are shown in Figure 12.

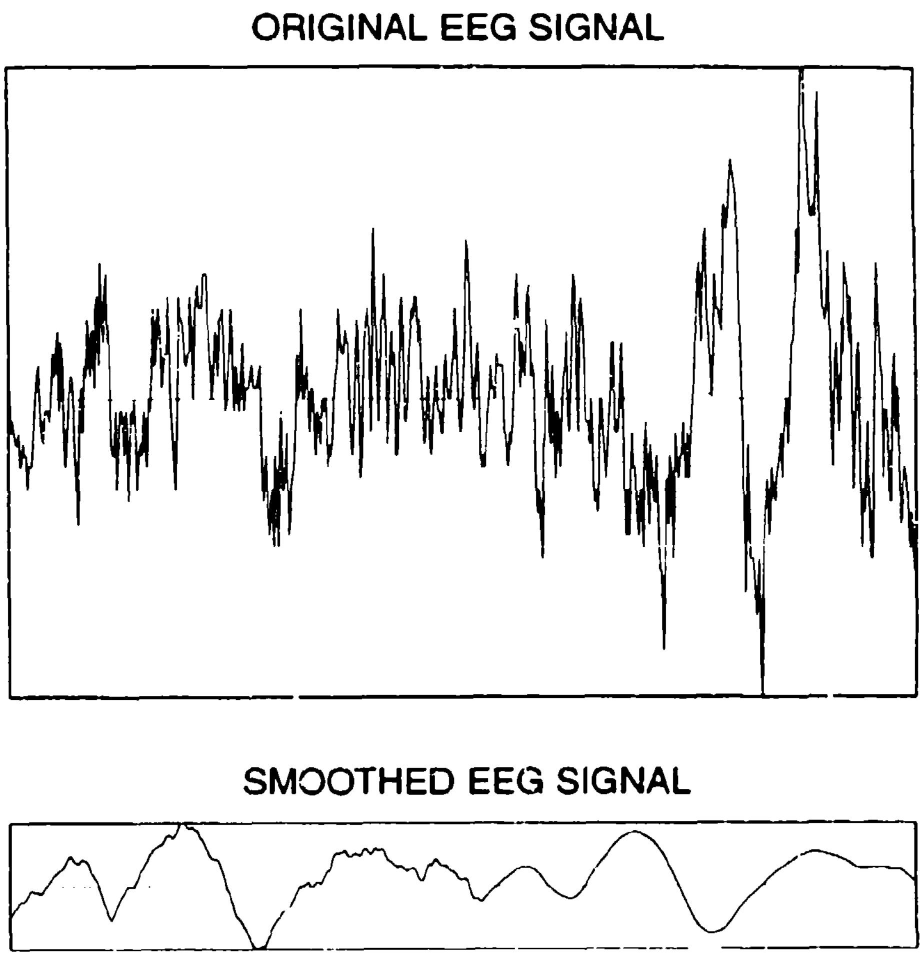


Figure 12. Advantage of a Smoothed EEG Signal. Note the multiple axis crossings in the original signal resulting from high energy, high frequency components. These crossings are absent in the smoothed version, leaving only the axis-crossings at the lower frequencies.

The smoothing algorithm used was:

$$y_j = \frac{1}{2n+1} \sum_{i=j-n}^{j+n} y_i \qquad (1)$$

where: $y_i$ = the amplitude at time (sample) $i$

$n$ = the radius of the smoothing window

The radius of the smoothing window was determined heuristically. An $n$-value of 14 was found to reduce the energy of high-frequency components sufficiently to uncover some of the lower frequency axis crossings of sample EEG signals. These smoothed signals were first substituted only for the reference signal in order to set the timing of the draw events in the display. However, this sacrified amplitude resolution when using the scaled intensity option since the reference signal determined the intensity of the bars in the display. The channel signals, which drove the displacements of the bars, continued to use the original EEG inputs.

The third modification was an option to skip over a selected number of reference signal draw events (peaks or axis-crossings). This lowered the effective frequency of the screen updates while maintaining some control link between the brain's actual activity and the display's output.

To provide another option in exploring the low frequency components of the EEG signals, the fourth modification caused the Phase II display to calculate the mean sample number between every two axis crossings of the reference signal. When these calculated sample numbers were treated as actual axis crossings, the effect was the same as doubling the axis-crossing frequency. During the testing, it was found that reference/channel relationships were more easily seen in a peak-driven display when the reference signal was a first-order harmonic of the channel signal, producing a 2-band display. It was hypothesized that by effectively halving the constantly modulating frequency of the reference signal in this manner, that a 2-band display might result which would be easier to detect than the 1-band display of 2 signals fundamentally related to each other. Tests with simple and summed sinusoids supported this approach, but it was not as helpful in EEG signals analysis. Because of this, and due to the time delays caused by the preprocessing to calculate the additional data points, this option was later abandoned.

The fifth modification was the addition of computer-generated low-frequency sinusoids as reference signal options. Complete data records could then be displayed and their relationships to 0.5, 1, 1.5, and 2-Hz signals studied.

Other options were also added for controlling the intensity of the plots on the display. Intensity scaling according to reference signal amplitude often seemed more distracting than helpful and so the user was given the alternatives of clamping the intensity to a maximum value or using a bi-level intensity scheme based on a user-selected threshold. This threshold option was intended to address the display distortions arising from the limited resolution of the digital input signals: signals crossing the axis will probably not be sampled at the zero-amplitude point but at some point very near the axis crossing with some non-zero amplitude. If the signal crosses the axis at some steep slope, as EEG signals do, the amplitude of the sample may be significantly displaced from zero. As a result, digitized test signals were producing ragged patterns using axis-crossing events which were not as easily recognized as the smooth patterns produced by computer-generated signals, which contained all of the exact axis-crossing values. Using this option, those bars resulting from a reference signal value which had changed sign from its predecessor, but which was greater than the selected threshold, was drawn in a subdued shade to deemphasize its presence. This acted to focus the viewer's attention on the bars in the overall pattern which resulted from reference signal values under the displacement threshold.

*5.3.3 User Interface.* Linking control over these many display options within the AFIT toposcope to user menus greatly simplified the task of viewing various data records repeatedly while varying display parameters. Descriptions of these menus and their associated functions appear in the *TOPOS User's Guide*, Appendix G.3.4.

## *5.4 Chapter Summary*

A computer program was written in two phases to implement the display methodology selected early in the research. TOPOS, the resulting AFIT toposcope, offers various display and processing options which were added as the tool matured during testing. These options allow the user to vary bar intensity and width in order to enhance the patterns present. Draw events can be selected in order to control the effective frequency of the reference signal and both fixed-frequency signals and EEG channels can be used as reference signals. The results of using TOPOS to analyze test signals and an EEG record of motion-sick individuals are the subject of the next chapter.

# *VI. Results: The Toposcope's Point of View*

Finally, the last task of this thesis was using TOPOS to study test signals and digitized EEG data. The test signal analysis was an important first step to attempt to characterize the toposcope's display for various signal frequency and phase relationships by using inputs with known characteristics. This would provide the basis for interpreting the more complex EEG displays and indicate possible problems in analyzing them.

## *6.1 Test Signal Analysis*

Test files were built in which each channel was a different signal; both periodic signals and EEG signals combined with a sinusoid were used. Appendix G.3.4 lists the test files' contents by channel. Using TOPOS, any channel could be referenced to any other channel, and the relationship displayed. Low-frequency sinusoids, both of constant and varying frequency, were generated by the digitization procedure used for the EEG signals, producing an off-frequency signal. They were also generated by computer program, producing true-frequency signals. Later, display performance was checked with EEG signals which had been summed with 1-Hz sinusoids of various amplitudes.

*6.1.1 TOPOS' Initial Configuration.* The time-shifted display was used because of its addition of a time axis to the display. Seventy "time bins" were used to provide the viewer with as much history of the patterns presented as the display would allow. Screen updates were driven by reference signal axis crossings. While peak-driven drawing was also a valid method for displaying test signal relationships, it was confusing with EEG signals because of their amplitude modulation and so was not used. Bar width was varied as necessary for clarity. Narrow bars were helpful in detecting patterns in complex displays, as were broad bars in simpler displays. Bar intensities were not varied using either of the developed options because the peak reference signal amplitudes they represented in one case were valid only in peak-driven TOPOS displays, and in the other case, their representation of sampling resolution error was deemed more confusing than helpful. Instead, bars were drawn using maximum intensity.

*6.1.2 Display Interpretation.* The displacement of the displayed bars from the central horizontal axis in each display was afforded the most importance and used as the sole feature in determining the signal's relationship to the reference and in search of any pattern presented across multiple channels. This was justified because the bars represented both the reference signal's frequency and the observed channel's relationship to it. The reference signal frequency was represented on the screen by the timing of each new bar's appearance simultaneously in every channel's display, and the displacement of each bar represented the scaled amplitude of that channel's signal at that instant in time. It was hypothesized that similar relationships between channel signals and the reference would produce similar displays. Likewise, dissimilar relationships would result in distinguishable patterns. The resulting patterns from the low-frequency sinusoids fell into four categories, based on the signals' relationship.

*6.1.2.1 Patterns of Fundamental Relationship.* When the reference channel and the observed channel were of matching frequencies, the TOPOS display showed a single, horizontal line in the observed channel. This pattern was a ragged approximation of the horizontal line if the input signals were not sampled very near their axis crossings, as was often the case with digitized signals. However, when the signals differed slightly in frequency, this single-line pattern split into two interlaced and phase-reversed sinusoids. Examples of these three patterns appear in Figure 13. In addition, Figure 14 shows how two signals of slightly differing frequencies result in the interlaced sinusoids using draw events triggered by reference signal axis crossings.

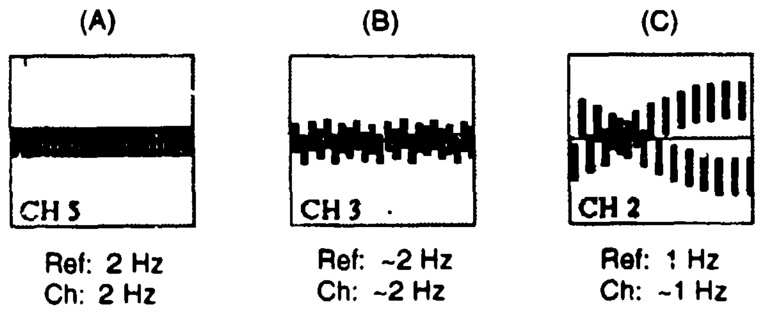


Figure 13. A TOPOS Display of Fundamental Relationships. Pattern A resulted from two signals of matching frequency. Pattern B was a ragged approximation of Pattern A due to the signal's matching frequency but insufficient sampling resolution. Pattern C arose when the two signals were nearly, but not exactly, the same frequency.

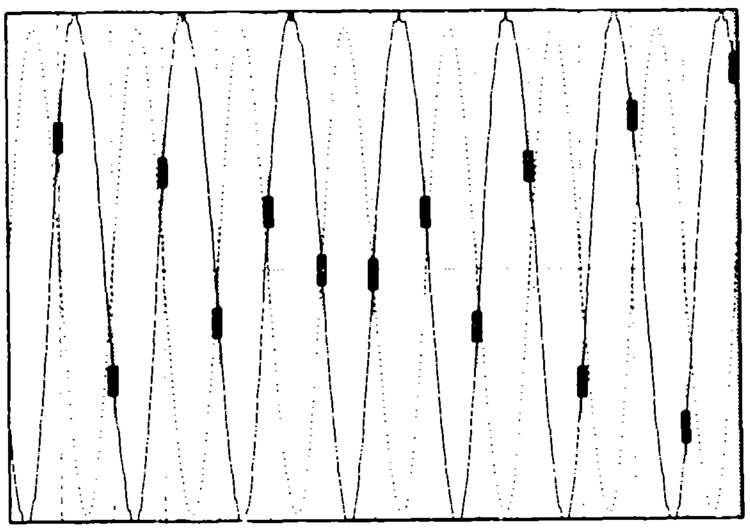


Figure 14. A TOPOS Display Point-by-Point. This figure provides insight into how and why two signals of approximately, but not exactly, the same frequency produce an interlaced, phase-reversed pattern on the screen. The reference signal is drawn with a dashes. Vertical lines mark the refererence signal axis crossings and boxes mark the observed channel signal's amplitude at the time of those crossings. These boxes, which trace the diverging portion of the overall pattern, are what appear on TOPOS' display.

*6.1.2.2 Patterns of Reference Signal Harmonics.* Three different patterns also emerged whenever the observed channel's frequency was an integer multiple (a harmonic) of the reference frequency. If the channel signal was a perfect harmonic of any order, the display was a single horizontal line, indistiguishable from that of the true fundamental frequency relationship. However, if the channel signal's frequency was only approximately an odd multiple of the reference, the interlaced pattern of near-fundamental relationship re-emerged. When, instead, the channel signal represented only an approximate, even multiple of the reference frequency, a single sinusoid appeared in that display, with its period inversely related to the frequency difference between channel and reference signals. These harmonic/near-harmonic patterns are shown in Fig 15.

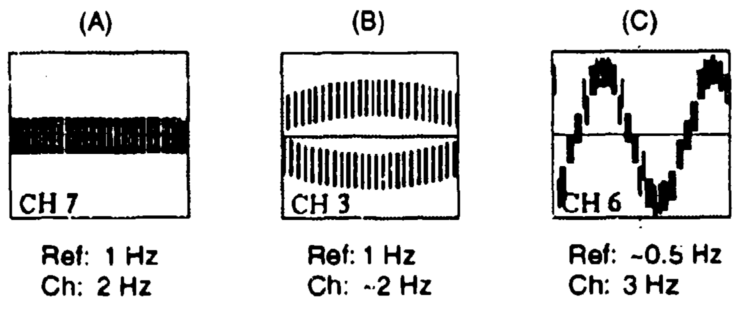


Figure 15. A TOPOS Display of Harmonic Relationships. Pattern A resulted from a signal channel which was an exact harmonic of the reference signal. If the channel signal was a near, but not exact, harmonic and was most nearly an odd multiple of the reference frequency, Pattern B was displayed. Pattern C arose when the channel signal was again off-frequency, but nearly an even multiple of the reference frequency.

*6.1.2.3 Patterns of Reference Signal Subharmonics.* Figure 16 presents another trio of patterns produced when the observed channel's frequency was exactly or approximately an integer factor (a subharmonic) of the reference frequency. A new pattern consisting of three horizontal lines was created by channel signals which were perfect first-order subharmonics of the reference signal. The sinusoidal pattern

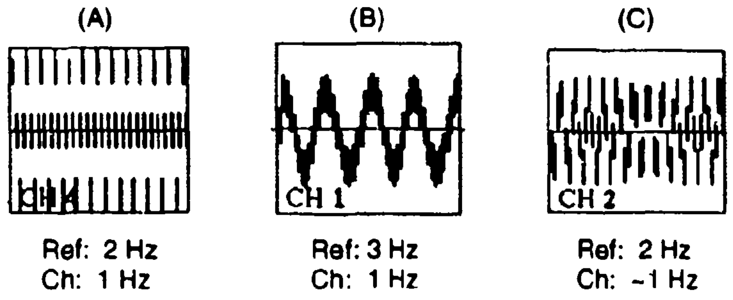


Figure 16. A TOPOS Display of Subharmonic Relationships. Pattern A resulted from a signal channel which was an exact first-order subharmonic of the reference signal. If the channel signal was approximately a second or higher-order subharmonic, Pattern B was displayed. Pattern C arose when the channel signal was again off-frequency, but nearly a first-order subharmonic of the reference frequency.

appeared again in the cases when channel signals were approximate second or higher-order subharmonics of the reference signal. Another new pattern consisting of four interlaced sinusoids grouped in two phase-reversed pairs resulted from signals which were approximate first-order subharmonics.

*6.1.2.4 Patterns of Phase Shift.* Several test signals in one data set were of matching frequencies but incorporated varying amounts of phase shift. The patterns they displayed when referenced to a common signal indicated three categories of phase shift effects. First, no change in display would result if the original and shifted waveform's axis crossings were indistiguishable.[1] Second, if the original signals produced one or more sinusoidal patterns on the screen, the phase shift caused a corresponding horizontal shift in the displayed pattern. Third, when the original pattern was one or more horizontal lines, the phase-shifted signal caused a vertical "spreading" of the pattern, initially increasing the number of horizontal lines

[1] For example, the amplitude at a specific time sample of a 4-Hz sinusoid after a 90-degree phase shift does not change from the sample of the original waveform at the same time. This is not the case, however, with a 3-Hz sinusoid.

and then increasing the distance between them. These last two types of phase-shift effects are shown in Figure 17.

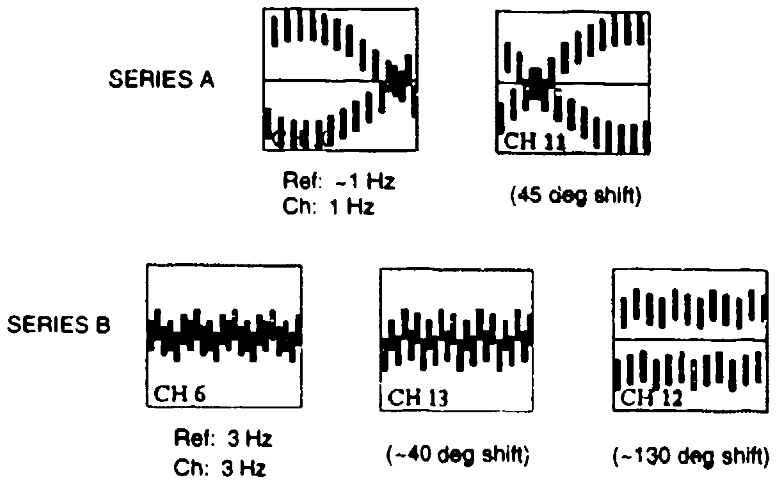


Figure 17. A TOPOS Display of Phase Shift. Series A shows the leftward shift in a sinusoidal pattern caused by the negative phase shifting of one signal. Series B shows the "spread" that occurs in a linear pattern.

*6.1.3 The Ambiguity Problem.* In the tests, nine categories of signal frequency relationships presented only five unique pattern types, and no one-to-one mapping existed by which each pattern could point to only one possible signal relationship. However, useful information was still available in the sense that a fundamental or harmonic/subharmonic relationship was signaled, even if the exact nature of that relationship was unknown. For example, when TOPOS displays a single-line pattern, it indicates that the input signals are of exactly the same frequency *or* that the observed channel signal is a perfect harmonic of the reference signal. Also, if the pattern is the two interlaced, phase-reversed sinusoids, the signals are near-fundamentals *or* the observed channel signal is an approximate, odd multiple of the reference frequency. This ambiguity had to be considered when interpreting the patterns resulting from TOPOS' analysis of EEG signals. To evaluate TOPOS' usefulness with EEG inputs, the test signals composed of EEG summed with a 1-Hz sinusoid were referenced to signals fundamentally, sub-harmonically, and harmonically related. The significant result was when the composite signal was referenced to a 1-Hz sinusoid. The pattern presented followed the same random pattern as that of the unaltered EEG when referenced to the 1-Hz signal, but was noticeably compressed towards the central horizontal axis in the display box. It was hypothesized that the complex sinusoidal patterns that were associated with subharmonic and harmonic relationships in the earlier tests were functions of the periodicity of the signals used and might not be presented during EEG analysis.

## *6.2 EEG Analysis*

*6.2.1 Preparations.* Prior to beginning the EEG analysis, an initial criteria for what constituted a significant pattern was needed and several more changes were required to TOPOS to better adapt it to the differences between the test signals and EEG inputs it would now process.

*6.2.1.1 Target Pattern.* It was decided to primarily look for compressions of the EEG-produced patterns as signals of a significant relationship with the reference signal. This would be most easily recognized as an attempt for the display to approximate the pattern presented if the reference signal was referenced to itself.[2] However, despite the outcome of the tests using the EEG/1-Hz composite signal, spreading of the pattern sinusoidally (interlaced sine waves) or linearly (multiple horizontal lines) might also occur and would be considered important indications as well.

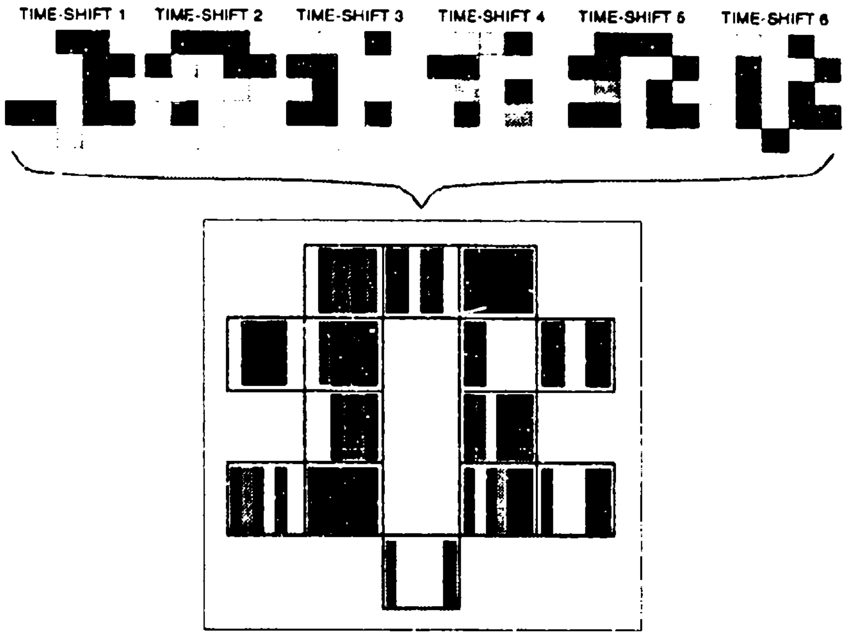
The TOPOS display would be monitored for simultaneous occurrence of either pattern: the compression or spread. The resulting cross-channel pattern could ap-

[2]This pattern is available on the TOPOS display if the selected reference signal is one of the 14 channels. However, if the reference signal is one of the four sinusoidal signals which are also options, no display is available for comparison. Nonetheless, the user should remember that the pattern would be that of a true fundamental relationship with no phase-shift: a vertically-centered straight line.

proximate the development described by Vogen and shown earlier in Figure 1. In addition, it was hypothesized that one channel would drive the cross-channel synchronization and when that channel was used as the reference signal, the activity pattern would be most easily visible.

*6.2.1.2 Recognition Problems.* The EEG records differed in three significant ways from the majority of the test signals used earlier. First, they were constantly and rapidly changing frequency under normal conditions. Second, the brain activity pattern which Vogen reported they held, was also reported to continually wax and wane. This meant that without time-averaging the data, as Vogen's FFT/contour-mapping method did, the sought-after pattern could be intermittent as well as different in shape from screen update to screen update. The third significant difference was that the EEG signals were also rapidly amplitude modulated, meaning that axis crossings would often be made at very high slopes and the x-axis-displacement resulting from the inadequate sampling resolution could be amplified. This would contribute to a lack of smoothness in any resulting pattern.

*6.2.1.3 Final Configuration.* Four additional modifications were made to the TOPOS configuration. The first was necessary because it quickly became apparent that the time-shifted display was presenting far too much information to support the detection of a pattern which might exist for only a few screen updates at a time due to an EEG's rapid frequency modulation. Therefore, the number of time bins displayed on the screen was made user selectable, permitting as few as 6 or as many as 74. The lower settings allowed the user to focus his or her attention on a region small enough to detect cross-channel patterns which were short-lived. Figure 18 shows the second change, the addition of a correlation grid in the center of the TOPOS screen. This was necessary to perform one additional processing step and better integrate for the user the information of TOPOS' 14 channel displays. The integration was required because it was difficult for the user to visually track the



**TOPOS Correlation Grid**

Figure 18. TOPOS' Correlation Grid. Each of the square regions corresponds by location to an EEG channel in the larger display. The six colored bars in each region represent three levels of similarity in that channel's displacement from the reference signal's amplitude in the six samples. Black indicates a very close relationship and white indicates large differences. The top row of smaller grids are how the display would look over time without the time-shifting.

patterns in all 14 channel displays simultaneously because of their spatial separation. The additional processing came in the form of the correlation grid showing each channel's degree of displacement, not from the x-axis as the rest of the TOPOS display was showing, but from the reference signal's amplitude at that sample. This was shown instantaneously in each region of the correlation grid in the same time-shifted format used in the channel displays and in three colors to indicate degrees of agreement between the signals.

The third modification resulted from the need to slow the display to permit the user to analyze the constantly changing presentation. Display update speed was inversely scaled to the number of time bins in use and options were added to momentarily pause the display, as well as to step it one screen update at a time, or one "batch" (a complete update of all time bins) at a time. The fourth modification did not involve software changes, but required the use of an option not found helpful before. It was necessary to use only every seventh axis crossing of the reference signal to trigger screen updates in order to lower the effective frequency into the region of interest (below 4 Hz) when using an EEG channel as reference.[3] Using only every third axis crossing was required even with a smoothed EEG signal as reference.

*6.2.2 TOPOS' Output.* Because of the complexity of the display resulting from EEG inputs and the length of the EEG record, an 11-second block of data was chosen for analysis. The data was from Vogen's Subject 5 and began at 4 minutes, 58 seconds into the record. This same block was singled out by Vogen as a prime example of peak motion-sickness brain activity (44:48-49,70). His thesis included 12 color contour maps of 6-second FFTs made at 0.5-second intervals (44:70).

[3]Because of the EEG's rapidly varying frequency, the instantaneous frequency calculated from inter-event times constantly varied and not all draw-events were accompanied by frequency readouts of less than 4 Hz.

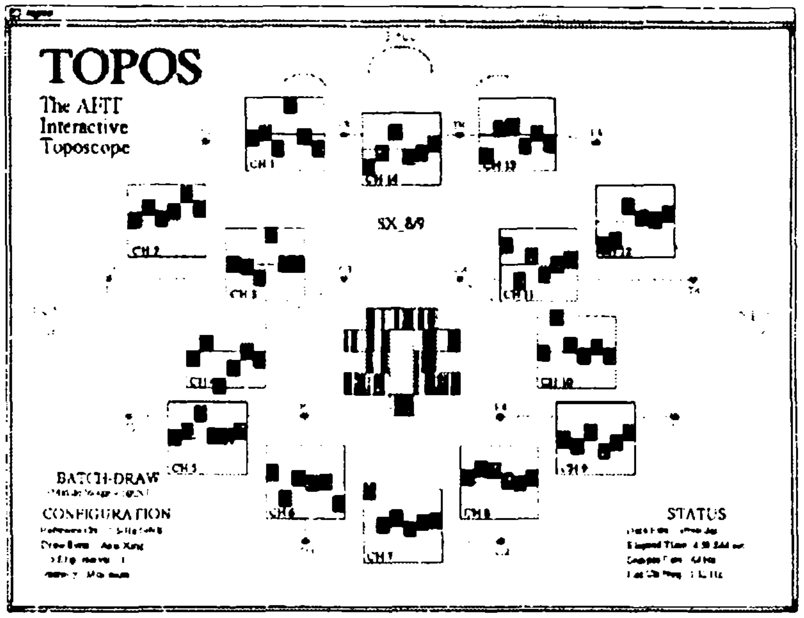


Figure 19. TOPOS' Display of EEG with 1.5-Hz Reference (Part 1).

*6.2.2.1 1.5-Hz Reference.* The 1.5-Hz sinusoid was selected as the initial reference signal because the Vogen contour maps of interest were displayed at that frequency. This TOPOS run was a check for how the EEG data block related to a fixed frequency which Vogen had considered optimal. Because the reference's frequency was in the delta region, no skipping of draw events was required to adjust the frequency.

The channel displays, provided in Figures 19–25, were only sometimes highly correlated[4] to the reference signal as the correlation grid changed rapidly. In addition, each channel showed frequent episodes of activity that was highly related to the reference signal for periods spanning 2 samples. However, at intervals during the data block, certain channels were very closely related to the reference signal for a larger number of samples. Figure 26 shows the location and frequency of highly related channel-reference activity as shown by 3 or more consecutive black bars on the

[4]As measured by the correlation grid.

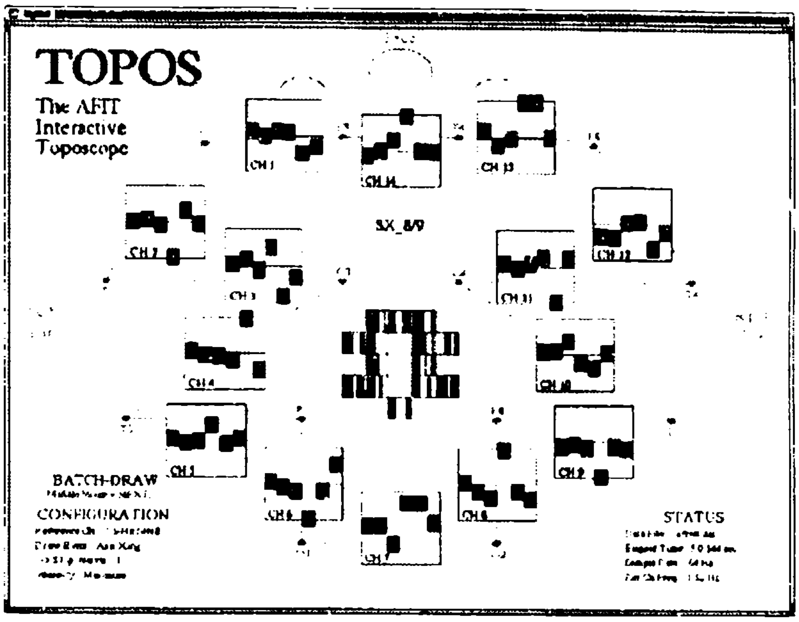


Figure 20. TOPOS' Display of EEG with 1.5-Hz Reference (Part 2).

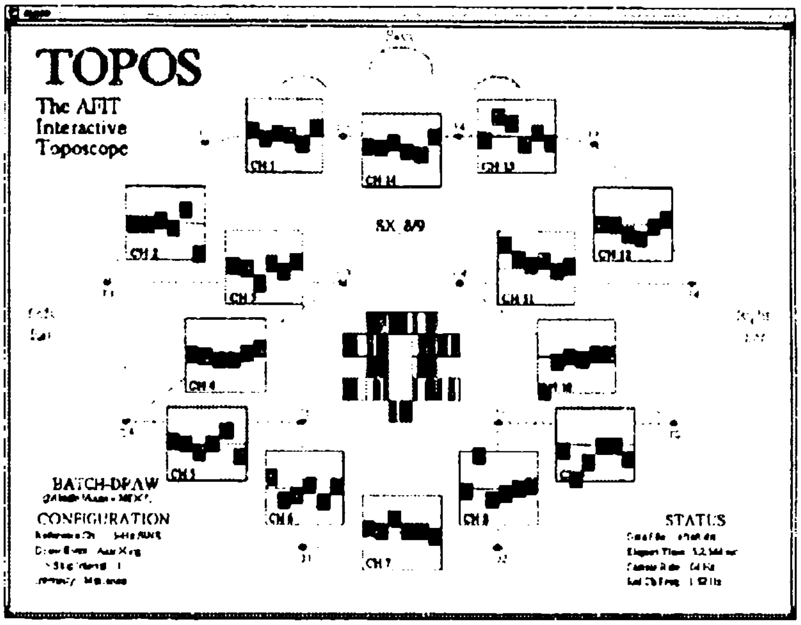


Figure 21. TOPOS' Display of EEG with 1.5-Hz Reference (Part 3).

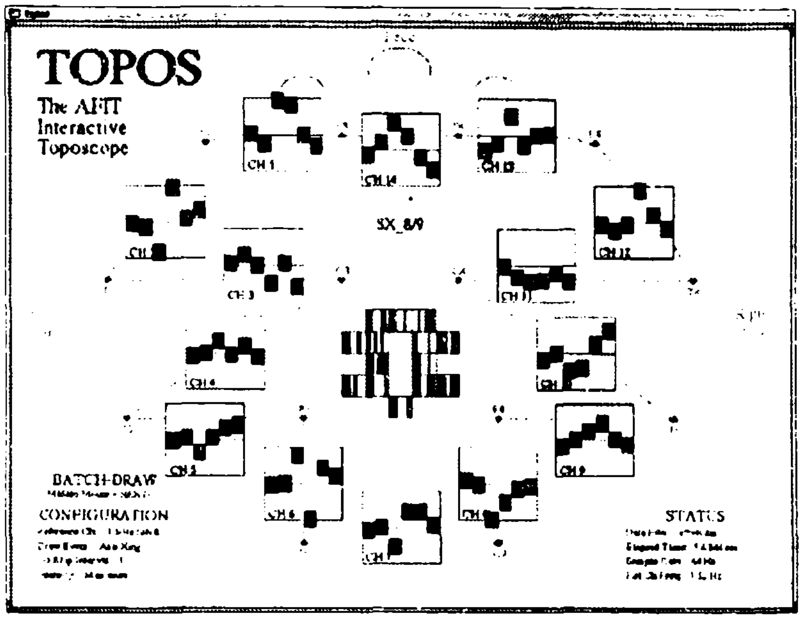
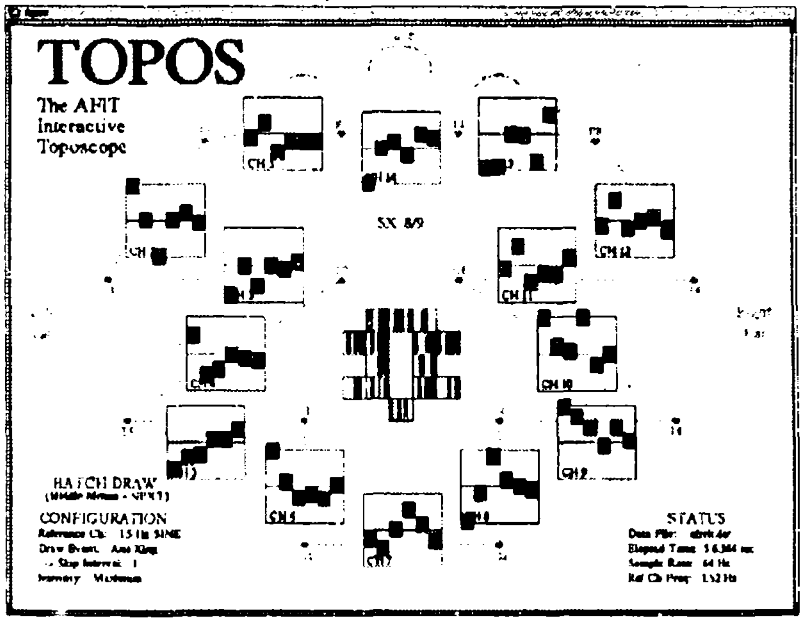


Figure 22. TOPOS' Display of EEG with 1 5-Hz Reference (Part 4).



CONFIGURATION
Reference Ch: 1.5 Hz SINE
Draw Event: Axe Xing
-> Skip Interval: 1
Intensity: Maximum

STATUS
Data File: [illegible]
Elapsed Time: 56,344 sec
Sample Rate: 64 Hz
Ref Ch Freq: 1.52 Hz

Figure 23. TOPOS' Display of EEG with 1.5-Hz Reference (Part 5).

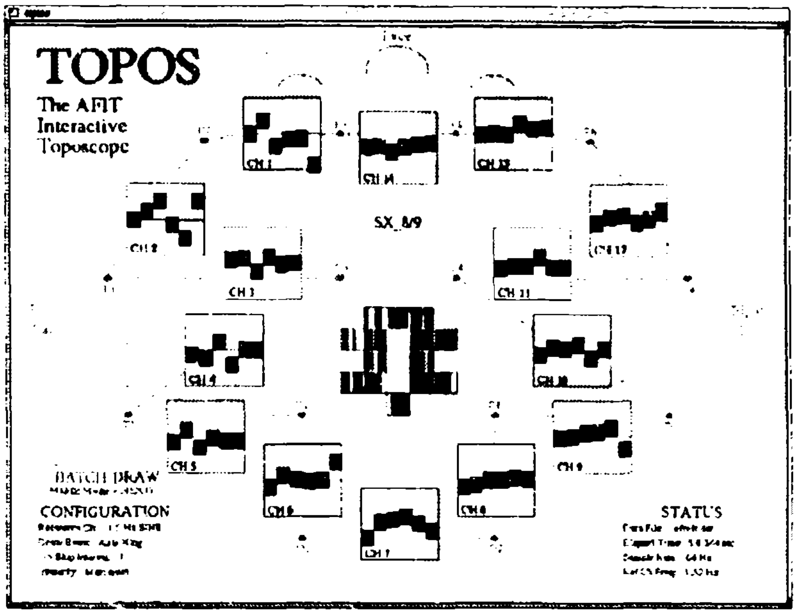


Figure 24. TOPOS' Display of EEG with 1.5-Hz Reference (Part 6).

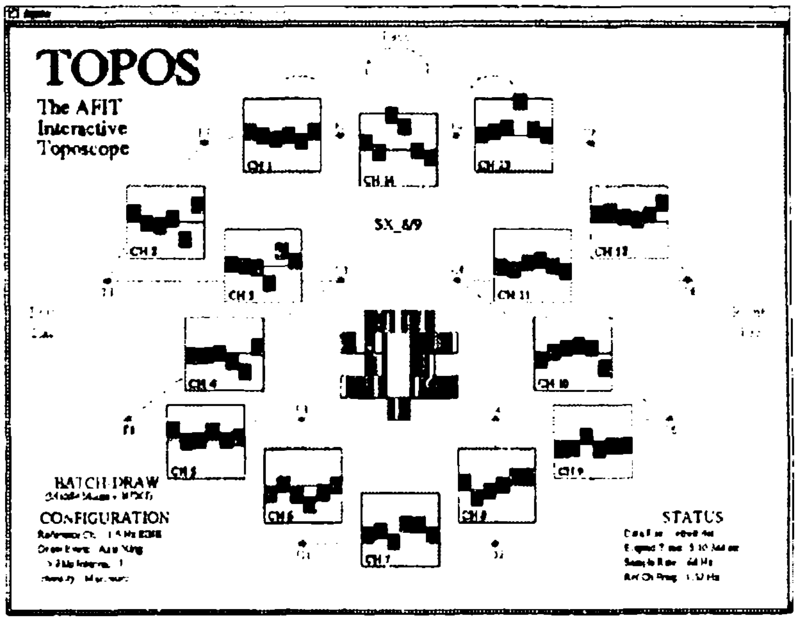


Figure 25. TOPOS' Display of EEG with 1.5-Hz Reference (Part 7).

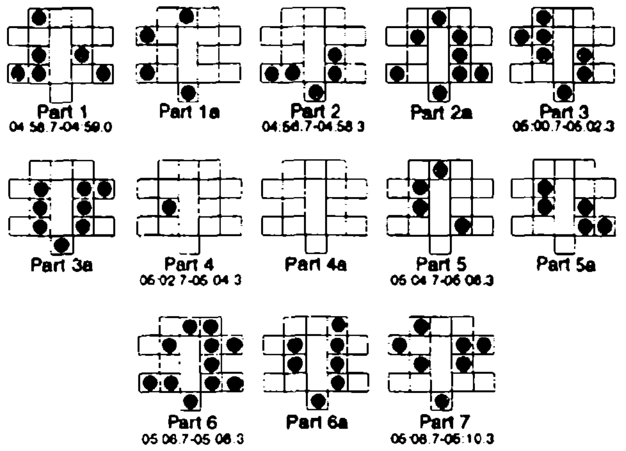


Figure 26. Pattern of High Correlations on the TOPOS Correlation Grid. Each black dot signifies three or more consecutive highly correlated amplitudes between that channel and the 1.5-Hz reference. Parts 1–7 are summaries of consecutive displays spanning the same data segment as Figures 19–25. The intermediate patterns, Parts 1a–7a, display high correlations which overlap between the original displays.

correlation grid. The diagram also accounts for the artificial breaks in the correlation grid due to the still-captures of the updating display.

The correlation mapping confirmed the constantly-shifting nature of high, channel-reference relativity with a 1.5-Hz reference signal. Also, the number of highly correlated channels waxed and waned until Displays 4 and 4a in Figure 26. At this point, the number dropped sharply, with the low occuring during samples at average data record times between 5:04.5 (min:sec) and 5:05, and then built back to a higher level in Displays 6–7. Although Vogen's maps were time-averaged, the sequence he provided shows the development of the brain activity pattern he reported. His pattern showed an average decline until a low was reached at approximately 5:05 before activity increased again. Thus the TOPOS display appeared to closely approximate the Vogen's pattern. However, Vogen's pattern did not then continue to increase, as did the correlation grids on TOPOS. Instead, the pattern repeated twice again the build-drop-build pattern in a rhythmic manner prior to the end of the data block. The TOPOS correlation grid did not support these final two drops, even when the criteria for meaningful correlation was raised to four or five consecutive black bars.

However, the single-sinusoid and "spread" patterns seen earlier in the test displays did occur in the TOPOS displays of the EEG record. Channel 14 in Figure 22 and Channel 5 in Figure 23 show the sinusoidal pattern and and Channel 11 in Figure 19 shows the convergence of a sinusoidal spread and Channel 3 in Figures 22 and 23 show both the divergence and convergence. Both of these patterns indicated a near-fundamental, subharmonic, or harmonic relationship between channel and reference signals. Furthermore, these more complex relationships are not considered in the simple comparisons display of the correlation grid.

*5.2.2.2 EEG Channel References.* Changing the reference to one of the EEG signals now changed the focus from the EEG's relationship to a constant frequency, to its correlation to one channel within the record. Also, the displays would also be compared with Vogen's map sequence. Three channels were selected, based on the locations of earliest and highest activity in Vogen's contour maps. Channel 6 was chosen as an example of the initial left-parietal activity, Channel 3 represented the activity in the left fronto-temporal region, and Channel 11 included the right temporal focus Vogen noted (44:48). Because the original, digitized EEG signals were used, only every seventh axis-crossing was displayed in order to lower the instantaneous reference frequencies into the 1–5 Hz range.

The resulting displays were analyzed for the three activity drops shown on Vogen's maps, as well as for any pattern of synchronization between the remaining channels and the reference. All three reflected the first drop in low-frequency brain activity, as did the TOPOS display using the 1.5-Hz reference. However, none of the EEG-referenced displays reflected the two later decreases.

Also, none of the displays appreciably increased their correlation levels over that of the 1.5-Hz reference. None of the three reference channels "captured the attention" of the other channels any more than another. Thus, no channel could be hypothesized to be driving the activity, much less have originated it.

*6.2.2.3 Smoothed References.* Smoothed EEG signals were used in two scenarios: 1) only the reference channel was smoothed, and 2) all the EEG signals were smoothed. The first scenario was to use smoothing as a means to obtain the lower frequency component of the reference signal without distorting the remaining channels. The second scenario was to obtain the lower frequency component while distorting all channels somewhat equally. These displays were evaluated similarly to those in the previous section and found to perform approximately the same. One difference was overall higher displacement correlation levels in the display using all smoothed inputs, probably due to the slow rise and fall of smooth curves.

*6.2.2.4 Other Observations.* The most disturbing observation was the lack of any increasing trend from low to high levels of correlation in the TOPOS display of EEG as Subject 5 progressed from sitting stationary without motion stimuli to severe motion sickness and emesis. While fluctuations were constant, the channel-reference correlation was *higher* at the start of the record when baseline EEG was being collected. This could be due to the lower amplitudes of the resting EEG in each channel being displayed on a scale determined by the standard deviation of the entire record.

It should also be noted that high correlation levels did not imply fundamental or near-fundamental relationships between channel and reference signals. Such relationships could have existed, but harmonics of the reference frequency would also produce the same pattern.

## *6.3 Chapter Summary*

The software base of TOPOS continued to play a key role in its usefulness as an analysis tool. As a result of studying the test signal displays and in anticipation of the different characteristics of EEG data, several modifications to TOPOS then preceded its application to a motion-sickness-affected EEG data block. These included the ability to vary the number of time bins in the time-shifted display and to control the rate of screen updates in order to better focus on short-lived patterns. Also, the correlation grid aided the observer by integrating in one location the instantaneous correlation levels of axis-crossings of channel and reference signals. Test signal analysis using TOPOS showed several categories of patterns that could be produced by various relationships between two signals. However, none of them could uniquely point to one of the relationships. A block of motion-sickness-affected EEG data studied earlier by Vogen was then studied using TOPOS and both fixed-frequency and EEG channel references. The TOPOS display did not show evidence of Vogen's pattern, did not identify any of three key channels as the pattern's driver, and did not show a trend of increasing channel-reference correlation as motion sickness symptoms began and worsened.

# *VII. Conclusion*

The last eight months of work have translated the Toposcope of Walter and Shipton into a multi-featured, software-based tool capable of translating frequency and phase relationships between two signals into a graphical display. All research objectives were met: the AFIT toposcope was designed and built, test signals were displayed and characterized, and motion-sickness-affected EEG data were analyzed.

The toposcope was chosen for this research because of the instantaneous, unembellished nature of its display. It offers signal analysis which does not depend on time-averaging and does not create an entire landscape of electrical activity from only a handful of data points. The most recent tool last used in AFIT's motion sickness research, the FFT-based contour brain map, had both of these characteristics. Thus, the toposcope offered a different point of view.

The design of TOPOS was evolutionary. A vertically travelling bar was selected over the rotating vector of Walter and Shipton to better facilitate coding, but the principle of their earlier devices was retained. That bar was chopped, shifted, narrowed, and enlarged as the display was changed to increase its usefulness. In its final configuration, TOPOS first notes the data samples in which one signal, the reference, crosses the x-axis. The tool then shows the user the degree of displacement from the x-axis of 14 different signals at those samples. There are many possible display patterns that result, depending on the offset between the axis crossings of the reference and displayed signals, and these offsets vary with the signals' instantaneous frequency and phase shift. The patterns appear on the screen in segments as the display updates are right-shifted until the display's boundary is reached. The user can visually compare the 14 channel displays for similarities to the reference signal's pattern or to other patterns which are indicative of sub-harmonic or harmonic relationships. Or, the user can watch the correlation grid at the center of the display for an indication of the degree of similarity between the channel and reference signals' displacement at each sample. A user's first impression of TOPOS' display is well characterized by Walter's description of his own toposcope: bewildering and difficult to interpret (46:62). Yet watching the display over a period of time, coupled with an understanding of how the displays results, leads quickly to more familiarity with the tool and the meaning of its output.

TOPOS offers users distinct advantages over its hardware-based predecessors. Because it is constructed in computer code, TOPOS gives users real-time control of display parameters in order to enhance and detect displayed patterns. Varying the number of time bins, the width and intensity of bars, and the draw events which are displayed can highlight the presence of a significant pattern for an observer. TOPOS also makes the computer an ally in helping the user to interpret the displays. The correlation grid in the TOPOS display is such an application of computer processing to bring together information for better assimilation.

Because it is a computer program, TOPOS works only with digital inputs. Some test signals and the available, analog EEG data were interfaced to the toposcope using the results of investigations which discovered the Brain Atlas digitization and file formats. Sinusoidal test inputs were used to explore the tool's indications of frequency and phase relationships between two signals. A single EEG file recorded during a severe motion-sickness episode was successfully interfaced to TOPOS. It was then studied for evidence of fundam[illegible]al, sub-harmonic, and harmonic relationships with an external, low-frequency signal, as well as between channels in the record.

As the test and analysis phase progressed and the resulting TOPOS displays were collected and studied, it became increasingly evident that TOPOS was not displaying the anticipated brain activity pattern in a form recognizable to observers. This raised two issues of significance. The first issue was the display's inability to unambiguously differentiate between any two significant frequency relationships. The common patterns shared by fundamentally related signals and certain harmonics, for example, reduced the usefulness of the tool. The second issue was the lack of any overall increase of channel-reference correlations in the motion-sickness-affected EEG record. These two issues begged the question "Why?"

In attempting to answer this inquiry, the tool's data-handling integrity was the first area considered. Soon after the first test signals were displayed on TOPOS, quality control became a regular concern. Internal computations and data manipulations, as well as the graphics they produced were thoroughly cross-checked.

The next area of investigation was TOPOS' design. An initial decision to adapt Walter and Shipton's circular format to a line-based display began the evolution of how TOPOS functioned. Design restrictions came and went, such as the peak-drawing mode, as the display changed to increase readability and usability. This usability came in both the form of user-control and the ability to focus on frequencies below 4 Hz. Central to the final display was the use of reference signal axis-crossings for frequency derivation. This method was also used by Shipton, but TOPOS did not offer a corresponding display of channel amplitudes as Shipton's display did, and the opportunity was created for confusion between signals of differing frequencies, but similar axis crossings. While the Walter-Shipton device effectively displayed the amplitude of channel activity between reference signal axis crossings, TOPOS "strobes" the channel signals and measures their amplitude only at the sample where the reference signal crosses the axis.

The third area of consideration was the tool's application. All of the Walter and Shipton references reviewed during this research addressed the utility of their Toposcope in the analysis of evoked potential EEGs. This application confined their attentions to extremely short periods of time following each stimulus provided to the subject, and when these stimuli were repeated regularly and rapidly, Walter and Shipton could observe the brain's response over and over again. In this research, however, the arrival of the various motion stimuli at the brain could not be timed, much less controlled, leaving the observer to look for "suspicious" patterns developing over long stretches of an EEG record. This led to the use of Vogen's contour brain maps as the focus of the EEG analysis. His maps spanned a small segment of EEG data and offered a position from which to begin the pattern search.

The final area of investigation was the validity of the brain activity pattern mapped by Vogen. This was never really questioned, and especially given the design and performance problems of TOPOS, should not be. A goal of this research had been to study motion-sickness-affected EEG data with the hope of observing the instantaneous development of the brain activity patterns and increasing the understanding of the phenomenon, not to disprove it.

While any user could benefit from more time to study the output of TOPOS, the design issues outlined above are considered to contribute the most to its limited success as a signal analysis tool. Its task of simplifying the confusing relationships in the human EEG is formidable, but achievable. In 1953, Grey Walter was addressing the complexity of the brain's electrical activity and he admonished, "...it would be foolish to expect the enchanted loom to weave a fustian fabric" (45:292). It would also be short-sighted to assume TOPOS to be incapable of displaying a more significant portion of that weave.

# Appendix A. *Preprocessing the Data*

This appendix highlights the knowledge gained from the EEG data records used as ir  to the toposcope. Fourteen data records existed in analog form on tape and in digital form, but in an unknown Brain Atlas format, on AFIT's Bio-logic computer. The goal was to either re-digitize the analog tapes or discover the format of the digital records, and then convert their contents into their original voltage values.

## *A.1 Re-Digitizing Options: NeXT and CODAS*

Given the unidentified format of the Brain Atlas data files and past difficulties in obtaining technical information on the program, the f :st point of investigation was the possibility of re-digitizing the records from the analog tapes. While the NeXT workstations in the AFIT signal processing lab could perform A-D conversions, they could only convert one channel at a time. This was a significant problem since the EEG records were 14 time-synched channels. If a NeXT was used, a series of synch pulses would have to have been recorded over each channel, each channel individually digitized into a separate file, and then the files merged together while simultaneously recovering the channel synchronization.

Another alternative was CODAS,[1] a signal processing package available on a personal computer (PC) in the motion sickness lab. This program could simultaneously digitize three to five channels using a special board installed in the CPU. However, this option would have still required the creation of multiple files and the resynchronization of the channels when recombined.

It was very important that all 14 EEG channels retain their time synchronization in order to ensure the validity of any brain activity pattern displayed by the

[1] A software product of Dataq Instruments, Inc.

toposcope. Therefore, before expending large amounts of time and effort to learn the synchronization techniques and gather the equipment required by the NeXT and CODAS options, it was decided to examine the Brain Atlas data files in search of insight into their storage format.

### *A.2 Brain Atlas*

The task of decoding the method which Bio-logic Systems Corporation used for data storage was made much easier when they mistakenly sent a copy of that format to AFIT. Because of past difficulties in obtaining information from the company, obtaining a copy of the file structure had previously been thought highly unlikely. However, typographical errors were discovered in the Bio-logic document. The following section will discuss some of the specifics of the Bio-logic EEG data files resulting from the study of Brain Atlas documentation and experimentation. The procedures referenced to the appendices are based on the Brain Atlas Users Guide (4) but were developed during the experimentation phase to improve on the user interface offered by the user guide.

*A.2.1 Bio-logic EEG Data File Structure.* The Brain Atlas software located on the Bio-logic PC in the Motion Sickness Lab is capable of simultaneously digitizing 14 channels of analog EEG data. However, the binary files it creates include six additional channels which it internally interpolates prior to data storage from those channels actually recorded. The names of these files follow the format "Exxxxxxx.DAT" to differentiate them from evoked potential (EP) data files, which are prefixed with the letter "F." Due to AFIT's nominal collection rate of 64 samples per second and average recording time of 20 minutes, the EEG data files range in size from 500 to 800 kilobytes. These are binary files using an MS-DOS format. The first 1920 bytes[2] are header information in a pattern common to both EEG and EP

[2]1 byte = 8 bits

files. Therefore, much of the header content is invalid for EEG applications. Those values which are important appear in Equation 2 on page 73 and are discussed in that section. Also, the header size of 1920 bytes is different than that shown in the Bio-logic documentation. The documentation errors found during the research are listed below.

- Header length — Header is 15 blocks[3] long, not 13 as printed (3:3).
- EEG Data Points — Begin with byte 1920.[4] Stored as 1-byte integer numbers indicating analog-to-digital (A-D) counts, not as "sums saved as three-byte integers" (3:10).
- Voltage Conversion Equation — Produces values with offsets by channel of 0 to -4 $\mu$volts from those used by the Brain Atlas.
- Header location of SENS and CALUV — Both are 1-byte (vs. 2-byte) integers beginning at the location shown in the Bio-logic format documentation.

The location of key EEG record parameters within the Bio-logic data files are provided in Table 1.

Table 1. Key to Brain Atlas Data File Header. All values are in hexadecimal. Notes: 1) Invert for approximate sample rate (either 64 or 128 Hz); 2) See Table 4 for location of each channel.

| FIELD | BYTES | DESCRIPTION |
|---|---|---|
| Sample Period | 416 | in msecs (Note 1) |
| Calibration Values (C.V.) | 425-438 | offset in A-D counts from +80h (Note 2) |
| D.C. Offsets (D.C.) | 446-459 | offset in A-D counts from +80h (Note 2) |
| Sensitivity (SENS) | 467 | std EEG parameter in $\mu V/mm$ |
| Calibration Voltage (CALUV) | 468 | in $\mu V$ at std SENS |

[3] 1 block = 128 bytes.

[4] Header contains 1920 bytes numbered 0 to 1919.

*A.2.1.1 EEG Data Points.* Within the body of the data file are a series of 1-byte, integer A-D counts representing instantaneous voltages recorded over the EEG channels. They range from 0h[5] to +0ffh. However, 080h is considered zero. Thus, A-D data values less than 080h represent negative numbers and the actual number of collected counts for any data point is found by subtracting the artificial zero level 080h from the binary value found in the data file. This offset is included in the voltage calculation shown in Equation 2.

Prior to display in a Brain Atlas plot or topographic map, Bio-logic states that these A-D counts are converted into voltages using the following equation (3:10):

$$Voltage_{\mu volts} = [(ADV - 80\text{h}) - (DC - 80\text{h})] \times (2 \times \frac{CALUV}{CV}) \times \frac{SENS}{10} \qquad (2)$$

where: $ADV$ = number of A-D counts

80h = the assigned zero-level

$DC$ = DC offset for each channel (from header)

$CALUV$ = size of calibration signal in microvolts (from header)

$CV$ = $CALUV$ in A-D counts

The results of this equation should be the values displayed anytime the "Voltage" option is selected within the Cursor Analysis function option of the Brain Atlas (see Appendix F), or when the program is tasked to provide an ASCII output of a subset of the collected data (see Appendix E). However, this equation produces values which are 0 to 4 $\mu$volts less than the values calculated and displayed by the

[5]The format 0xh denotes hexadecimal.

Table 2. Offsets of Bio-logic Voltage Conversion Equation. Differences are between calculated values and values produced by the Brain Atlas and are in $\mu$volts.

| Ch | Offset | Ch | Offset |
|---|---|---|---|
| 1 | -2 | 8 | -4 |
| 2 | -2 | 9 | -1.98 |
| 3 | +0 | 10 | -4 |
| 4 | -2 | 11 | -1.98 |
| 5 | -2 | 12 | -2 |
| 6 | -4 | 13 | -2 |
| 7 | +0 | 14 | -2 |

Brain Atlas. These offsets for the recorded channels are constant across time but vary by channel as shown in Table 2.

The offsets for interpolated channels were not tracked. The reason for these anomalies was not found, but a documentation error is possible, given the incidence of errors as previously noted.

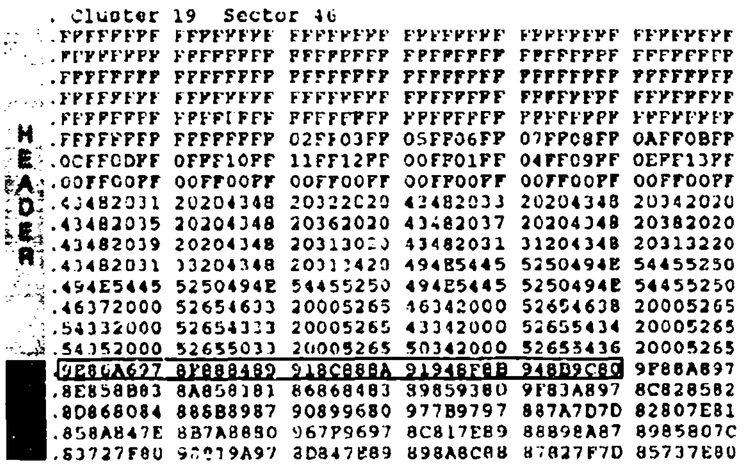
*A.2.1.2 Uncovering the EEG Data Format.* Using Norton Utilities, the MS-DOS-format Bio-logic data files were output in their hexadecimal notation and used to verify the location and validity of key header data as shown on the Bio-logic format sheet. The documentation was correct and showed the 21 1-byte values[6] of CV contained in bytes 425–445 of the file header. Values of DC offsets by channel were located in bytes 446–466. SENS and CALUV are actually 1-byte integers (vs. 2-byte, as published) beginning at bytes 467 and 469, respectively.

There is an error in Bio-logic's description of how the EEG data points were stored in the body of the data file (1). The points are not recorded as 3-byte, integer sums, but as 1-byte integers representing the number of A-D counts from

[6]Although the Brain Atlas only stores and displays 20 channels, the format for header parameters contains locations for 21 values.

each channel and for each sample obtained. The number of samples depends on the update rate selected and the length of the recording.

Port' ns of the data files were output using the ASCII Conversion option in the Bank Math function of the Brain Atlas (1). This tool only converts 256 data points, but using the procedure detailed in Appendix E, ASCII values of voltages could be produced. Yet the question of the location of each channel's value within the block of 20 or 21 data points (see Figure 27) for a single sample was still unanswered. For example, was Channel 1 the first data point or located elsewhere?



```
. Cluster 19  Sector 46
.FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
.FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
.FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
.FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
.FFFFFFFF FFFFIFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
.FFFFFFFF FFFFFFFF 02FF03FF 05FF06FF 07FF08FF 0AFF0BFF
.0CFF0DFF 0FFF10FF 11FF12FF 00FF01FF 04FF09FF 0EFF13FF
.00FF00FF 00FF00FF 00FF00FF 00FF00FF 00FF00FF 00FF00FF
.43482031 20204348 20322C20 43482033 20204348 20342020
.43482035 20204348 20362020 43482037 20204348 20382020
.43482039 20204348 20313020 43482031 31204348 20313220
.43482031 33204348 20313420 494E5445 5250494E 54455250
.494E5445 5250494E 54455250 494E5445 5250494E 54455250
.46372000 52654633 20005265 46342000 52654638 20005265
.54332000 52654333 20005265 43342000 52655434 20005265
.54352000 52655033 20005265 50342000 52655436 20005265
.9E8CA697 8F888489 918C888A 91948F8B 948D9C8D 9F88A897
.8E858B83 8A858181 86868483 89859380 9F83A897 8C828582
.8D868084 888B8987 90899680 977B9797 887A7D7D 82807E81
.858A847E 8B7A8890 967F9697 8C817E89 88898A87 8985807C
.83727F80 92919A97 8D847E89 898A8C88 87827F7D 85737E80
```

Figure 27. Hexadecimal Representation of a Brain Atlas EEG Data File Segment. The boxed numbers are one sample of 20 channels of EEG data (6 channels are interpolated).

This was especially difficult since the Brain Atlas signal junction box labeled the channel inputs numerically, but the Brain Atlas screen used alphanumeric designations. In addition, the Brain Atlas assumed monopolar recordings, which use a common reference for all electrodes, while AFIT had made bipolar recordings which used neighboring electrodes as recording references. This difference further confused the channel assignment search.

Table 3. Correlation Between Numeric, AFIT, and Bio-logic Channel Labels.

| Channel | AFIT | Brain Atlas |
|---|---|---|
| 1 | F3-F7 | F7 |
| 2 | F7-T3 | F3 |
| 3 | T3-C3 | F4 |
| 4 | C3-T5 | F8 |
| 5 | T5-P3 | T3 |
| 6 | P3-O1 | C3 |
| 7 | O1-C2 | C4 |
| 8 | O2-P4 | T4 |
| 9 | P4-T6 | T5 |
| 10 | T6-C4 | P3 |
| 11 | C4-T4 | P4 |
| 12 | T4-F8 | T6 |
| 13 | F8-F4 | O1 |
| 14 | F4-F3 | O2 |

To determine the channel locations within the data block, the Brain Atlas was calibrated using the procedure in Appendix C and a 200 mV square wave was collected following the procedure in Appendix D. EEG data was played over all but the one channel containing the square wave and 14 separate files were recorded, each with the square wave located on a different channel. The initial values of each file were then converted to ASCII and the values correlated with those displayed using the "Voltage" option in the Brain Atlas. This was possible because the square wave maintained a constant value across enough samples to clearly indicate the match between input channel and Brain Atlas electrode position. The resulting assignments of input channel number and AFIT and Brain Atlas electrode locations are shown in Table 3.

Determining these assignments made it possible to define the format of data storage and locate all the necessary values to input into Equation 2 in order to convert the A-D counts into voltages. The channel assignment within the DC offset

Table 4. Channel Locations in a Brain Atlas Header Block and ASCII File (*** indicates locations of interpolated values).

| | | | | | | |
|---|---|---|---|---|---|---|
| *** | *** | *** | Ch 01 | Ch 02 | *** | Ch 03 |
| Ch 04 | Ch 05 | Ch 06 | *** | Ch 07 | Ch 08 | Ch 09 |
| Ch 10 | *** | Ch 11 | Ch 12 | Ch 13 | *** | Ch 14 |

Table 5. Channel Locations in a Single Sample From a Brain Atlas Binary File (*** indicates locations of interpolated values).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *** | *** | Ch 01 | Ch 02 | *** | Ch 03 | Ch 04 | Ch 05 | Ch 06 | *** |
| Ch 07 | Ch 08 | Ch 09 | Ch 10 | *** | Ch 11 | Ch 12 | Ch 13 | Ch 14 | *** |

and CV blocks in the header is the same as in the Brain-Atlas-generated ASCII file and is shown in Table 4. The channel assignments for the EEG data samples are contained in Table 5. Using these data locations, the Equation 2 could be used to calculate voltages and determine the offsets in Table 2.

### *A.3 Reading EEG Data Files*

This knowledge of data locations and conversion calculations provided all the required input to process the EEG data within a Bio-logic file. The process involved transferring the data to the computer to be used during the programming, writing the code, and testing and validating the program and its output.

*A.3.1 Transferring the Data.* It was anticipated that a format conversion might be required before the MS-DOS binary data could be correctly read by the Unix-based Silicon Graphics Iris/4D workstation at AFIT. However, no conversion was necessary. The EEG file was transferred from the Bio-logic PC to a network PC in the AFIT Motion Sickness Lab using a data diskette. From there, the Silicon

Graphics workstation was accessed using the File Transfer Protocol (FTP) application and the file transferred in binary mode.

*A.3.2 The Data Preprocessing Program.* The goal of this programming effort was to calculate the EEG channel voltages from the data contained in the file. The first step was to produce a table of the values from the file header which are required in the voltage conversion equation. These values, along with the channel samples from the body of the file, could then be used to calculate the voltages of each channel for each sample in the input file. These tables were created early in the programming effort to use in ensuring that the data was being handled correctly. A portion of the results are in Appendix B, but the code was not preserved in the form used to produce the tables. Instead, it was converted into functions which output their results for graphical interpretation on the display.

Since each data point was stored in a 1-byte location in memory, they were read from the Brain Atlas file as C `char` (character) data types since they also require one byte each (40:A-4). The data points were then converted to integers prior to use in calculations. Required values from the file header (e.g., calibration voltages) were read into arrays and then used to convert each EEG sample from A-D counts into microvolts prior to writing it to the appropriate output file. No correction was made for the offset between the calculated value and the Brain Atlas value (see Table 2). Fourteen separate output files were created, each holding data from one EEG channel. Data from each channel were also analyzed to determine the approximate distribution, with a maximum, minimum, and standard deviation for each channel written to a single header file. Within the toposcope code, the function `prep_data()` performs data preprocessing.

## *A.4 Summary*

The conversion between analog EEG signals, binary Brain Atlas data files, and C-program output quickly became routine and dependable. This was possible due to the discovery of the Brain Atlas file format and the flexibility of the C programming language.

# Appendix B. *Data Obtained from Brain Atlas Data File*

```
CV Values by Channel:

BYTE   CH   INTEGER   HEX
----   --   -------   ---

 428    1     200     c8
 429    2     200     c8
 431    3     200     c8
 432    4     200     c8
 433    5     200     c8
 434    6     200     c8
 436    7     198     c6
 437    8     200     c8
 438    9     202     ca
 439   10     200     c8
 441   11     202     ca
 442   12     200     c8
 443   13     200     c8
 445   14     200     c8


DC Offsets by Channel:

BYTE   CH   INTEGER   HEX
----   --   -------   ---

 449    1     129     81
 450    2     129     81
 452    3     128     80
 453    4     129     81
 454    5     129     81
 455    6     130     82
 457    7     128     80
 458    8     130     82
 459    9     129     81
 460   10     129     81
 462   11     129     81
 463   12      129     81
 464   13      129     81
 466   14      129     81

SENS = 10

CALUV = 100

               A-D  A-D
SAMPLE   CH   Hex  Dec   microV
------   --   ---  ---   ------


****SAMPLE 1****TIME 0.016****
   1      1    97   151   22.000000
   1      2    88   136   7.000000
   1      3    8d   141   13.000000
   1      4    85   133   4.000000
   1      5    88   136   7.000000
   1      6    87   135   5.000000
   1      7    8b   139   11.111111
   1      8    85   133   3.000000
   1      9    84   132   2.970297
   1     10    87   135   6.000000
   1     11    80   128   -0.990099
   1     12    8c   140   11.000000
   1     13    81   129   0.000000
   1     14    8a   138   9.000000

****SAMPLE 2****TIME 0.031****
   2      1    97   151   22.000000
   2      2    81   129   0.000000
   2      3    85   133   5.000000
   2      4    81   129   0.000000
   2      5    84   132   3.000000
   2      6    85   133   3.000000
   2      7    88   136   8.080808
   2      8    88   136   6.000000
   2      9    88   136   6.930693
   2     10    8b   139   10.000000
```

# Appendix C. *Brain Atlas Data Collection Calibration*

PURPOSE: *To prepare the Brain Atlas (BA) program for data collection. Not required if you have not exited the program since last calibration.*

*STEP 1:* At the `C:>` prompt, type `BA` and press [ENTER].

*STEP 2:* Select the CPU turbo mode.

ACTION: Press [ALT]-[CNTL]-[-]("minus" on keypad).

STATUS: Lights the "Fast Clock" LED on front of CPU.

*STEP 3:* Select "EEG Collect" on Main menu.

*STEP 4:* Select "Calib" on Collection menu.

*STEP 5:* Create 400mV 1-Hz square wave using a function generator and o-scope.

*STEP 6:* Set GEN MODE on function generator to TRIG.

STATUS: Oscilloscope shows flat trace.

CAUTION: *Make A-D port connection carefully to avoid shorting the A-D board in the CPU.*

*STEP 7:* Route calibration signal to A-D port on CPU using 25-pin connector.

*STEP 8:* Select "Start Calibration" and wait for the beep.

*STEP 9:* Approximatelu 0.5 secs after the beep, turn GEN MODE switch on function generator to CONT.

*STEP 10:* Count 4 wavelengths of square wave and then return GEN MODE to TRIG.

If BA indicates "Calibration out of limits," do Steps 11-13.

*STEP 11:* Press [ESC] and note calibration wave forms collected.

*STEP 12:* Adjust timing of GEN MODE switch positioning on subsequent attempts to fill the time window displayed.

*STEP 13:* Repeat Steps 7-10 as needed.

If BA indicates successful calibration, continue.

*STEP 14:* Remove 25-pin connector from A-D port on CPU.

CAUTION: *Exiting the BA program deletes this calibration!*

# Appendix D. *Brain Atlas Data Collection*

PURPOSE: *To digitize analog EEG data or test signals and create DOS binary files ("Exxxxxxx.dat") in the Brain Atlas (BA) format.*

*STEP 1:* Ensure the BA program is calibrated (see Calibration Procedure).

*STEP 2:* Select the following in sequence:

"EEG Collect" on Main menu.

"Hardware" on Colection menu.

"Sample Rate" on pull-down menu.

"64 Hz" on option menu.

*STEP 3:* Turn on Disking function and open new data file.

ACTION: Select "File" on Collection menu.

Select "Disking" on pull-down menu.

Enter 7-digit file name (without "E" prefix or ".dat" extension) and select "OK."

STATUS: `DSK ON` flag on right side of screen.

*STEP 4:* Connect Bio-logic BNC channel input box to A-D port on CPU.

NOTE: *Data collection does not begin until approximately 2 seconds after "Start" is selected.*

*STEP 5:* Initiate input signal (e.g., turn on recorder).

*STEP 6:* Select "Start" on Collection menu (2 second delay).

*STEP 7:* To pause data collection, press [ESC] or right mouse button once.

*STEP 8:* To stop collection, press [ESC] or right mouse button twice.

*STEP 9:* Disconnect input cable from A-D port on CPU.

## Appendix E. *Brain Atlas ASCII Conversion*

PURPOSE: *Converts the first 256 samples of digitized Brain Atlas (BA) data into ASCII format. Data must first be stored in a BA "bank."*

NOTE: *The Cursor Analysis bank data option is not included here (see BA manual, page EEG-191).*

*STEP 1:* At the `C:>` prompt, type `BA` and press [ENTER]/

*STEP 2:* Select the CPU turbo mode.

ACTION: Press [ALT]-[CNTL]-[-] ("minus" on keypad).

STATUS: Lights the "Fast Clock" LED on front of CPU.

*STEP 3:* Select the following in sequence:

"EEG Analysis" on Main menu.

"File" on Analysis menu.

"Disking" on pull-down menu.

*STEP 4:* Select desired drive and data file.

*STEP 5:* Select "Banks" on Analysis menu.

NOTE: *Attempting to allocate more than one bank can result in insufficient memory for later operations.*

*STEP 6:* Select "Bank Data" on pulldown menu.

*STEP 7:* Select "1" on top-line option menu to bank data.

STATUS: None provided.

*STEP 8:* Press [ESC] or right mouse button to exit to Main menu.

*STEP 9:* Select the following in sequence.

"Map Analysis" on Main menu.

"Process" on Analysis menu.

"Bank Math" on pull-down menu.

*STEP 10:* Tab to the following fields, make entries, and press [ENTER]:

`Source bank:` <u>1</u>

`Destination bank:` <u>A</u>

*STEP 11:* Press [SHIFT]-[F3] to transfer data into a Bank Math letter bank.

STATUS: Lines at top of screen indicate "A" bank is allocated.

*STEP 12:* Back tab to the following fields, make entries and press [ENTER]:

`Source bank:` <u>A</u>

<u>NOTE</u>: *Failing to deallocate the letter bank will result in insufficient memory for later operations.*

*STEP 14:* Press [F8] to deallocate the letter bank.

*STEP 15:* Press [ESC] or right mouse button to exit Bank Math.

<u>To view the ASCII file contents, continue.</u>

*STEP 16:* Select "File" on the Analysis menu.

*STEP 17:* Select "Temp Exit" on the pull-down menu.

*STEP 18:* Type **`type txxxxxxx.asc | more`** and press [ENTER].

*STEP 19:* Press [ENTER] to view subsequent pages.

*STEP 20:* At `C:>` prompt, type **`exit`** and press [ENTER] to return to BA.

## Appendix F. *Brain Atlas Cursor Analysis of Data*

PURPOSE: *To display instantaneous voltages of all channels in a Brain Atlas (BA) data file. Also provides an interpolated brain map based on those voltages.*

*STEP 1:* At the C:> prompt, type **BA** and press [ENTER].

*STEP 2:* Select the CPU turbo mode.

ACTION: Press [ALT]-[CNTL]-[-] ("minus" on keypad).

STATUS: Lights the "Fast Clock" LED on front of CPU.

*STEP 3:* Select the following in sequence:

"EEG Analysis" on Main menu.

"File" on Analysis menu.

"Disking" on pull-down menu.

*STEP 4:* Select desired drive and data file.

*STEP 5:* Select the following in sequence:

"Display" on Analysis menu.

"Speed" on pull-down menu.

"Double" on option menu to display all data pts.

"Montage" on Analysis menu.

"Define" on pull-down menu.

"Select Group" on User Montage menu.

"I" on pull-down menu.

"Select Montage" on User Montage menu.

"0" on pull-down menu.

*STEP 6:* Backout to Analysis menu using [ESC] or right mouse button.

*STEP 7:* Select "Cursor" on Analysis menu.

To vary the scale of displayed plots, continue.

*STEP 8:* Select "Display" on the Cursor menu.

*STEP* Select "Raise" or "Lower" on the pull-down menu, as needed.

NOTE: *Start time of displayed data is located at the bottom-left corner of the screen.*

To move to earlier/later portion of data by entering start time, continue.

*STEP 10:* Select "Timer" on Cursor menu.

*STEP 11:* Select "Set to hh:mm:ss" on Set Timer menu.

*STEP 12:* Enter desired start time and select "OK."

NOTE: *Fine adjustment of cursor position is via the straight-arrow icons at bottom-right of the screen. Time of selected data point is displayed at bottom-left corner.*

To move to earlier/later portions of data one screen at a time, continue.

*STEP 13:* Select Z-shaped arrow icons at bottom-right of the screen for the desired direction.

To view a brain map of interpolated voltages, continue.

*STEP 14:* Position cursor by clicking the right mouse button at the desired point in the displayed plot.

To view instantaneous voltages in each data channel, continue.

*STEP 15:* Position cursor by clicking the right mouse button at the desired point in the displayed plot.

*STEP 16:* Select "Voltage" on Cursor menu.

NOTE: *Screen cannot be printed while "Voltage" display is present.*

To print the screen, continue.

*STEP 17:* Select the printer icon at the bottom-right corner of the screen

# Appendix G. *TOPOS User's and Programmer's Guide*

## *G.1 Introduction*

This guide is designed for two classes of readers: the user and the student or researcher who wishes to become familiar with the code's design in order to make modifications. Design notes will be mostly high level and intended to be supplemented by the body of the thesis report it accompanies, as well as the TOPOS code included in Appendix I. While the program behind the tool was written with great pride and care, it is not the product of an experienced programmer, so leniency and empathy is requested from all future reviewers.

## *G.2 Why TOPOS?*

This tool belongs in a class of spatial/temporal signal analyzers called *toposcopes* and is designed for the study of human brain electrical activity. The first toposcope used for similar purposes was conceived and constructed by Grey Walter and Harold Shipton in the late 1940s. TOPOS is an application of the principles of that original device. Prior to the advent of high-speed computers and digital processing, Walter and Shipton's Toposcope offered researchers a picture of brain activity based on frequency commonalities in spatially separate locations of the brain. These commonalities led to their early reports of functional links between neuron groupings in the brain as they worked together to process a visual stimulus.

Almost 45 years later, AFIT redeveloped this tool on a computer workstation to study brain activity patterns resulting not from a visual stimulus, but from the stimuli from multiple sources accompanying motion sickness. The frequency band below 4 Hz was the region of interest, and so several of the options are specifically designed to focus the tool on those frequencies.

As a toposcope, TOPOS offers an instantaneous viewpoint of frequency relationships between separate EEG electrode locations. While Fast Fourier Transforms (FFTs) and contour maps are popular methods of processing and presenting the same information, a toposcope does so without the time averaging required by FFTs or the interpolation of contour maps.

## *G.3 Using TOPOS*

*G.3.1 Prerequisites.* The program requires the following files to run:

topos

sinetest.dat.halfhz

sinetest.dat.1hz

sinetest.dat.1_5hz

sinetest.dat.2hz

an input data file in either Brain Atlas format

*or*

the 14 "*datafile*.chXX" files with accompanying "*datafile*.h" file.

Attempting to start the tool without all of these files will result in an error message to the console.

*G.3.2 Starting the Tool.* TOPOS is started by typing its name, followed by the data file name as shown:

```
> topos inputfile
```

If the input file is a raw Brain Atlas data file, its full name should be used. If the TOPOS *Prepare Data* option has already been run on the file, the name of the raw data file should still be used, since TOPOS will append the necessary file extensions before accessing the files.

*G.3.3 Help Screen.* Upon successful startup, a help/instruction screen is displayed which provides the program's default settings of display options and overviews the following mouse button functions.

| | |
|---|---|
| Left Mouse: | Halts display |
| Middle Mouse: | Continuous Drawing - Press pauses, release restarts |
| | Step Drawing - Press/release both draw next step |
| | Batch Drawing - Press draw next batch (redraws all bins) |
| Right Mouse: | Main menu |

The meaning of each type of drawing mode is explained below.

*G.3.4 TOPOS Control.* User menus greatly simplify the task of viewing a data record repeatedly while varying display parameters. The one control option not located on a menu is Display Pause. To momentarily pause the display when Step-Draw and Batch-Draw are not active, the user should depress the middle mouse button and hold it. Releasing this button restarts the display at the point it was paused.

All of the following menu options are available on the main menu. Many of the descriptions below also include comments on usefulness.

- *Prepare Data* — Selected only the first time that a new EEG data record is to be displayed. Data preprocessing, and the delay it requires, can be bypassed for that data record from then on. Subsequent sessions with a previously processed data file uses the separate channel output files as input streams.
- *Reference Channel* — Permits the online selection of any of the 14 channels or one of four sinusoids as the reference signal. Choosing a channel will cause TOPOS to display relationships of all channels to the instantaneous frequency (constantly *varying*) of that reference channel. Selecting a sinusoid as a reference results in a display of all channels as they relate to a *fixed* frequency.

- *Bar Width* — Permits selection of a narrow to broad bar pattern in the display. User should vary to their taste. Broader bars are helpful in simpler patterns and narrower bars in complex displays.
- *Bar Intensity* — Offers three options: scale to reference signal value, clamp to maximum value, or threshold at one of three reference signal values. Amplitude information provided by scaled option is of limited value and can be confusing when trying to ascertain a pattern. Thresholding can be helpful in simplifying . ıttern identification.
- *Display Mode* — Offers various combinations of single-bar/time-shift displays and continuous/peak/axis-crossing draw event triggers. Single-bar option is entertaining, but rarely useful. Peak drawing is fine with constant-amplitude test signals. but complicates the pattern for EEG signals (peaks of which vary widely in amplitude). Axis-crossing drawing is best choice.
- *Draw Interval* — Permits the modification of draw event intervals by skipping every n*th* event. Useful in getting down to lower frequencies of both smoothed and unsmoothed EEG data. Watch "Reference Signal Frequency" status line on screen to determine best selection.
- *Bin Number* -- Sets the number of bins used for time-shifted display. A low bin number is the key to displaying recognizable patterns in complex signals. Helps the user to look at the trees instead of the forest. Also note that the program is currently set to slow the display as the bin number is decreased. This permits the user to better view the displays presented.
- *Start Display* -- Allows the program to be started or restarted at the beginning of the data, at a specified time mark, or at the point where the display was last paused during that session. *This option is required anytime the screen shows "DISPLAY HALTED" over the status box.*

- *Test* — Controls the test modes of single-buffer drawing, step-drawing (draw one time-shift bin and wait), and batch-drawing (draw one set of bins and wait). Single-buffer drawing is the way around the "flicker" plaguing the TOPOS display. It is a "must" when using the lower bin numbers for time-shift drawing. These displays update much more slowly and double-buffer drawing is not suitable. Step drawing is useful when the user wants to control the display of each draw-event with the middle mouse button. Batch drawing prevents the overwrite of any bins prior to the user's signal via the middle mouse button. This is especially helpful when capturing images from the screen.
- *Help* — Brings up the help/instruction screen.
- *Swap Buffers* — Continuously swaps the two display buffers to enable the observer to see the points drawn on both planes whenever the display is paused.

*G.3.5 On-Screen Information* User information appears in three different locations on the screen.

- *Status Display* — In the lower-right corner, shows the data file being displayed, the sampling rate it was recorded at, the elapsed time[1] since the start of the data display, and the instantaneous reference signal frequency, as calculated from the number of samples occurring between draw events. Other messages, such as "Searching," "Display Paused," and "Display Halted" appear in this region as well.
- *Configuration Display* — In the lower left corner, shows the data channel being used as the reference signal for all 14 channel displays, the draw event interval trigger (axis crossing or signal peak), any adjustments to that interval (skipping), and whether the intensity of the bars is scaled to the reference signal,

[1]A calculated value based on the original sampling rate and the number of samples displayed up to that point.

clamped to a maximum value, or thresholded at a displayed reference signal value.

- *Data File Comments* — In the center of the screen, shows the comments manually added to the *data.h* file at the times specified. See Section G.4.2.3 for further information.

## *G.4 Programming TOPOS*

*G.4.1 Prerequisites.* The program requires the following files to compile:

topos.c
topos.h
tdraw.c
tmenu.c
tcontrol.c
tevent.c
tevent.h

The makefile included in Appendix I.7 turns these files into the executable code **topos**. Each program file is a functional grouping of functions coded in Silicon Graphic's implementation[2] of C and was executed on their IRIS-4D workstation without parallel processing optimization.

*G.4.2 Program Flow.* While the functional relationships in the TOPOS program could be derived from the code in Appendix I, this section and the flow chart in Appendix H are intended to provide a familiarization of how the program operates. Figure 28 is a very high-level diagram of the relationships within TOPOS.

*G.4.2.1 Initialization.* Invocation calls the function `main()` which initializes the TOPOS window and graphics and displays the initial screen before hand-

[2]The C language used by Silicon Graphics is based on the language as defined by Kerney and Ritchie's 1978 text, *The C Programming Language*, (40:A-1).

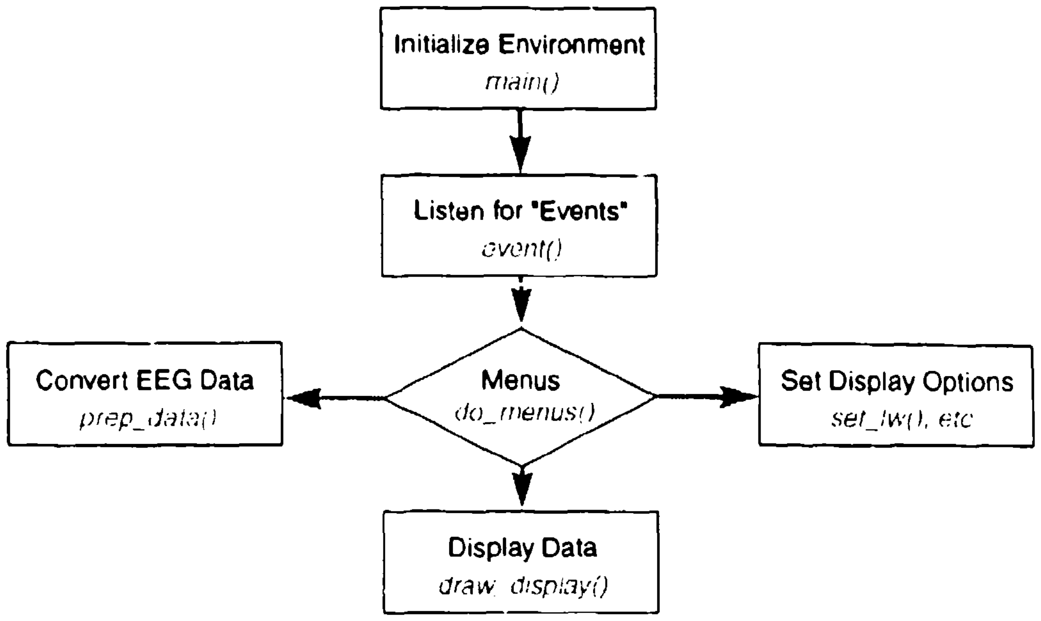


Figure 28. Functional Relationships Within TOPOS. Primary function names appear in italics. A detailed flow chart is in Appendix H.

ing over control to the function **event()**. This function constantly "listens" for mouse-clicks and other events and then performs the task assigned to that event. An example is the pop-up main menu that appears when the right mouse button is depressed.

The main menu is drawn by **do_menus()** and is the gateway to the remainder of the program. Each menu option calls a new function, sometimes passing a parameter as well.

*G.4.2.2 Display Options.* All of the display parameters listed in Section G.3.4 are controlled by separate functions, as shown in the non-exhaustive list below. The program default settings are determined in **tcontrol.c**. These functions assign numerical and/or boolean states to specific variable which are then used in the draw function of TOPOS.

- **set_b()** — Assigns the reference channel number to the variable **b** (brightness driver).
- **set_lw()** — Assigns the selected number of pixels to the variable **lw** (line width).
- **set_bins()** — Assigns the selected number of bins to use in the time-shifted display.
- **set_intensity()** — Determines if the displayed bar intensity will be constant or scaled to the reference signal's amplitude.
- **change_display()** — Determines whether the display will be time-shifted, if reference channel events will trigger draw function calls, and whether those events will be peaks or axis crossings.
- **zero_ptrs() / leave_ptrs() / positions_ptrs()** — Controls the point in the input data file where the program will begin to read in data
- **set_interval_skip()** — Controls the interval between draw events by assigning the number of reference channel peaks or axis crossings to skip and by assigning whether the midpoints between such signal events will also trigger draw events, effectively halving the time between events.
- **set_test()** — Turns on and off the test modes of single buffer drawing, step drawing, and batch drawing.
- **swap()** — Causes the back and front display buffers to swap continuously until an event occurs, such as a mouse click.

*G.4.2.3 Converting EEG Data.* Selecting the menu option "Prepare Data" starts the function **prep_data()**, which in turn calls other functions such as **read_hdr()** and **read_ch()** to implement the techniques discussed in Appendix A for converting the Brain Atlas EEG files into a form usable by TOPOS. The output is written to separate files, one for each EEG channel as well as one containing the maximum, minimum and standard deviation of amplitude values of each channel. This latter file has the same name as the input file, but with an additional "`.h`" extension.[3] The remaining output files are marked similarly with "`.chXX`" extensions, corresponding to their channel number.

*G.4.2.4 Displaying Data.* After initially drawing the display's background, the function **`draw_display()`** is a loop with multiple conditional statements. The execution of this function is controlled by the software switches provided by the functions in Section G.4.2.2 as **`draw_display()`** works through the data files one sample at a time. Because there are numerous possible combination of switch settings, there are many possible paths through this function. However, only a single software path which was used often during testing is presented below. The description provided will cover the more basic components common to most paths within **`draw_display()`**, as well as the functions of the specific switch options involved.

| | |
|---|---|
| TOPOS Configuration: | Reference CH: Ch 2 |
| | Bar Width: 5 pixels |
| | Bar Intensity: Scaled |
| | Display: Time-shifted Q axis crossings |
| | Bin Number: 10 |
| | Test Modes: Off |
| | Start: Beginning of Data |

1. Loop from first sample in the data files.
2. Break loop if mouse click event has occurred.
3. Find successive data samples in Channel 2 which change signs. Remainder of the function will use the first value of this data pair, corresponding to sample *s-1* in the program.

[3]Data file comments, if they are desired on-screen, must be manually added to this file following the channel parameters and in three fields: 1) minute time-tag (integer); 2) seconds time-tag (float); 3) comment (alpha-numeric, no spaces, less than 21 characters).

4. Swap display buffers to bring current display to the back for editing and to show the most recently drawn display.
5. **erase_bars()** — Erase the bar segment previously drawn in the time bin corresponding to the next draw event.
6. **draw_boxes()** / **draw_status()** — Redraw the channel boxes and configuration and status boxes.
7. **draw_bars()** — Set line width to 5 pixels.
8. **draw_bars()** — As required, clip any channel value at sample *s-1* to three standard deviations above the channel's mean.
9. **draw_bars()** — Calculate the vertical displacement of the bars to be drawn from each channel's value at sample *s-1*, scaling the maximum possible displacement to three times the standard deviations of each channel.
10. **draw_bars()** — Calculate the start and stop coordinates of the bar being drawn, time-shifted to the right from the previously drawn bar.
11. **draw_bars()** — Set the draw color to the gray-scale value calculated from Ch 2's amplitude at sample *s-1*, scaling the maximum possible displacement to three times the standard deviation of each channel.
12. **draw_bars()** — Draw a 5-pixel-wide line (the "bar").
13. **draw_bars()** — Repeat loop until interrupted or at end of data. Reset bin number to 0 every time Bin 9 is drawn.

*G.4.2.5 Obtaining Screen Prints.* Screen images of TOPOS displays were captured using the Silicon Graphics tool **snapshot** in a raster scan format. These were first converted to run-length encoded format using the program **tourt** and then to encapsulated postscript files using **rle2eps**.

Table 6. TOPOS Test File Contents by Channel.

| CH | `newtest.dat` | `vartest.dat` |
|---|---|---|
| 1 | 0.5 Hz | 1 Hz |
| 2 | 1 Hz | 2 Hz |
| 3 | 2 Hz | 3 Hz |
| 4 | 1 Hz | 4 Hz |
| 5 | 2 Hz | 1-4 Hz (varying) |
| 6 | 3 Hz | Sum (CHs 1-4) |
| 7 | 4 Hz | Sum (CHs 1,2,4) |
| 8 | Sum (CHs 1-3) | Sum (CHs 1,2,4,5) |
| 9 | Sum (CHs 4-6) | Sum (EEG, CH 5) |
| 10 | CH 4 with 45 deg phase-shift | Sum (EEG, $\frac{CH5}{2}$ |
| 11 | CH 4 with 90 deg phase-shift | Sum (EEG, CH 1) |
| 12 | CH 6 with 40 deg phase-shift | Sum (EEG, $\frac{CH1}{2}$ |
| 13 | CH 6 with 90 deg phase-shift | Sum (EEG, $\frac{CH1}{5}$ |
| 14 | 1.5 Hz | EEG |

*G.4.3 Test and Diagnostic Files.* Table 6 shows the content of each test file used during program development and included with the executable and source code upon thesis completion.

In addition, a diagnostic file listing is generated for every TOPOS session. The file's name is the data file with a `.diag` extension. It includes information regarding each draw event: sample number, displacements by channel, and other parameters.

*G.4.4 Program Bugs.* The following are known problems which the creator either did not know how or had insufficient time to overcome.

- User loses control over program execution if the window is repositioned, resized, or collapsed to an icon while the display is not showing "Display Halted" over the Status box. Requires user to kill process from console.
- All data display is erased when window is resized or redrawn (i.e., if user displays another application over it and then quits or moves that new appli-

cation's window). However, data file pointers have not moved and display can be restarted using any of the three "Start Display" menu options.

- No more than 10 comments can be read from the *data.h* file for display during TOPOS execution. If fewer comments are desired, then large times and empty comment fields should pad out the required 10 lines.

# Appendix H. *TOPOS Flow Chart*

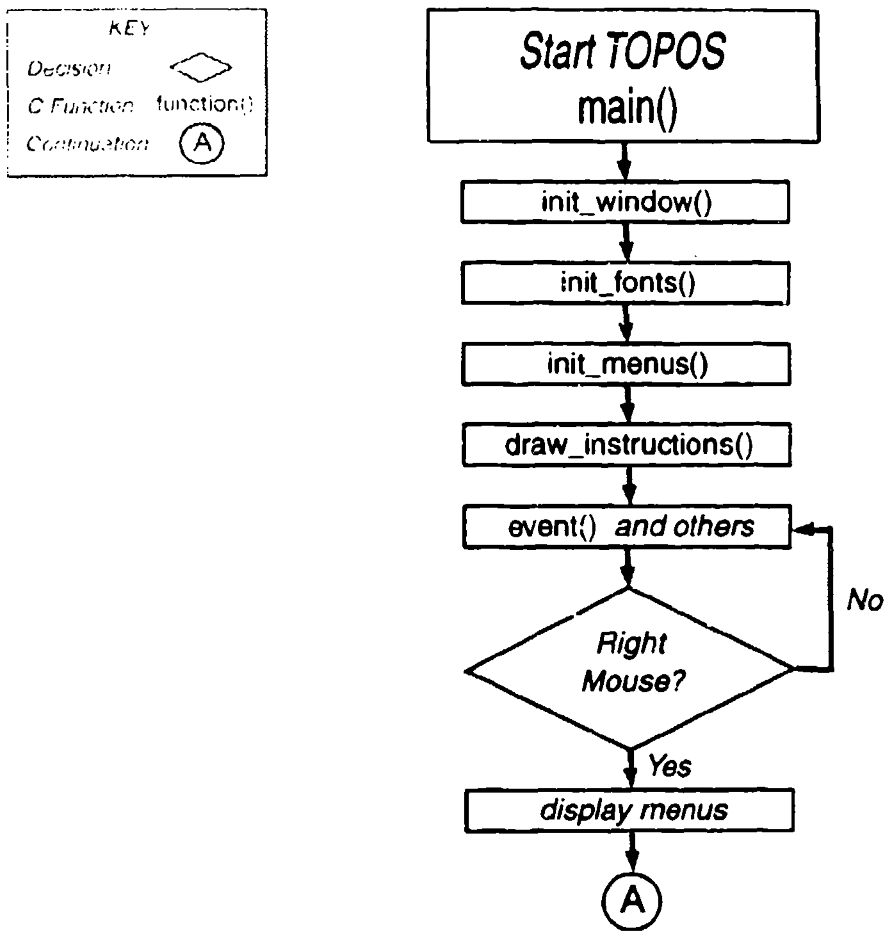


Figure 29. TOPOS Flow Diagram (Part 1 of 4)

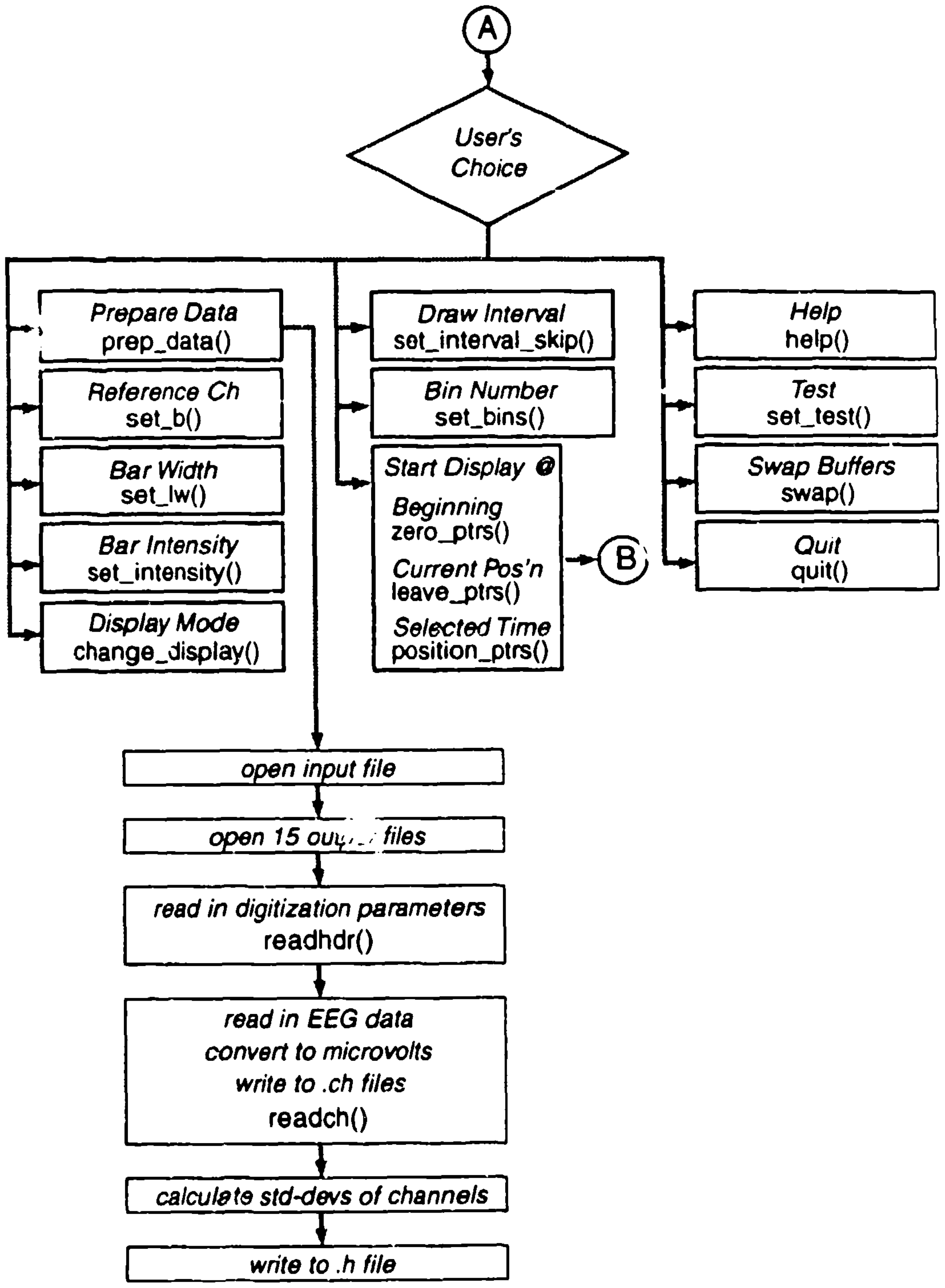


Figure 30. TOPOS Flow Diagram (Part 2 of 4)

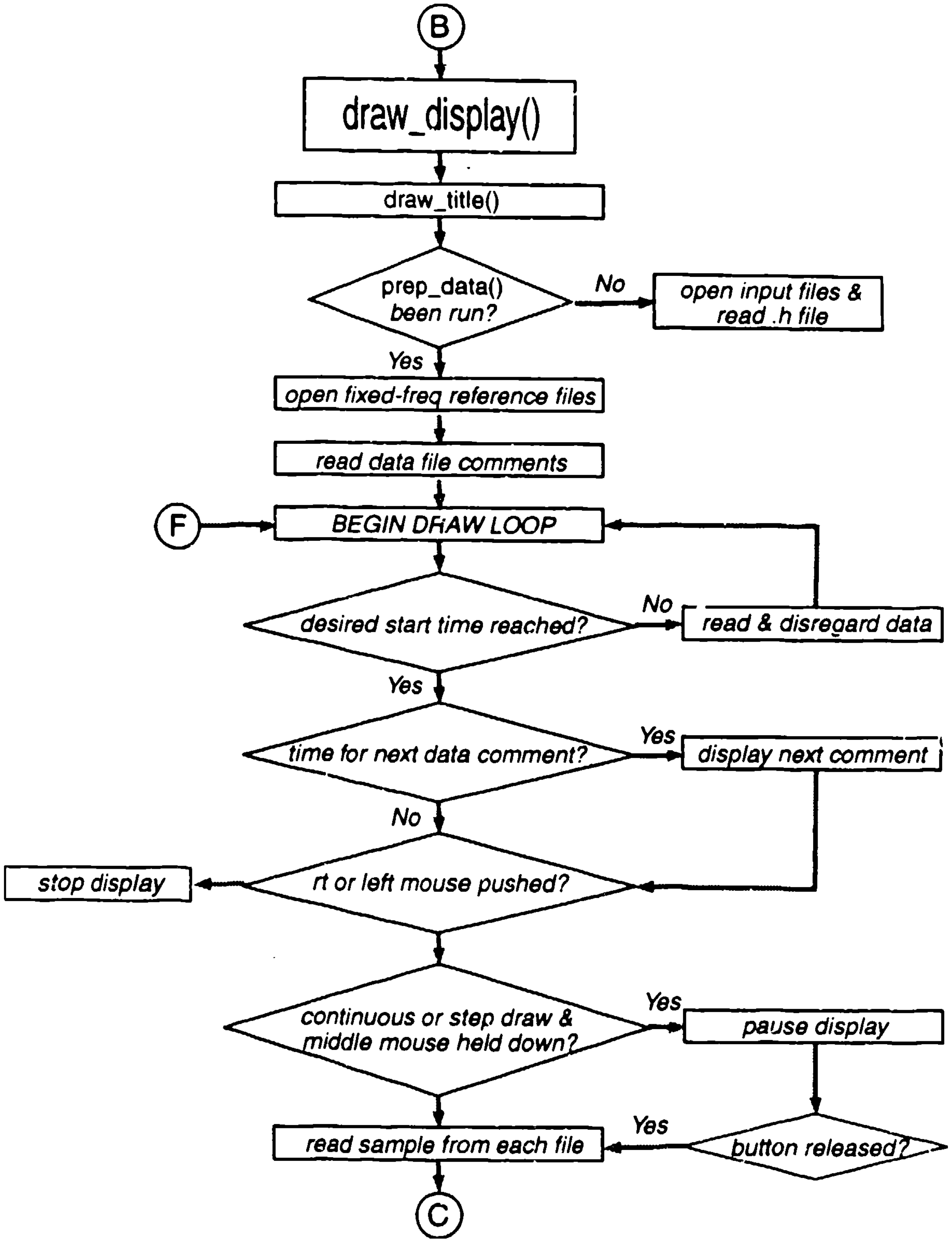


Figure 31. TOPOS Flow Diagram (Part 3 of 4)

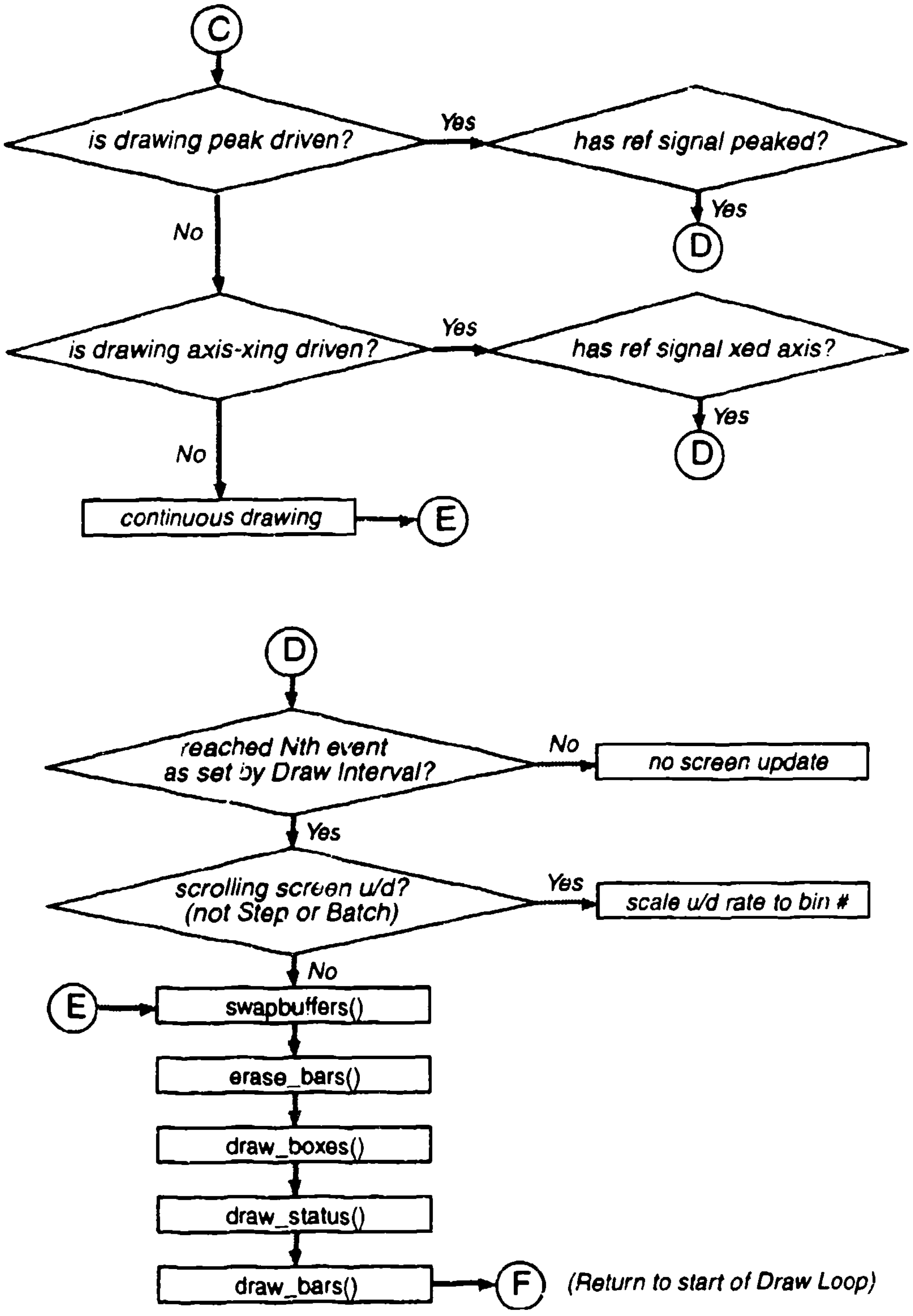


Figure 32. TOPOS Flow Diagram (Part 4 of 4)

# Appendix I. *TOPOS Code*

## *I.1 Main File: topos.c*

```
/*
 * topos.c
 *
 * The main file for the AFIT toposcope TOPOS.
 *
 * Capt Dwight Roblyer, AFIT/GSO-92D, November 1992
 *
 * Built over the program "CURVE DEMO"
 * by Howard Look for Silicon Graphics, June 1989
 *
 */


#include <stdio.h>
#include <string.h>
#include <gl.h>
#include <device.h>
#include <math.h>
#include <fmclient.h>
#include "tevent.h"
#include "topos.h"

/****  GLOBAL VARIABLE DECLARATIONS  ****/

FILE *ifp;         /* pointer to ".dat" input file       */
FILE *ofp[19];     /*            ".dat.chXX" files       */
FILE *hfp;         /*            ".dat.h" file           */

char AD,           /* holding var for read-in hdr data                  */
     caluv,        /* calibration volt in microvolts (from hdr)         */
     sens,         /* sensitivity  (from hdr)                           */
     bb,           /* bit bucket var for unwanted data  (from hdr)      */
     cv[15],       /* cal volts in A-D counts by ch (from hdr)  */
     dc[15],       /* DC offsets by channel (from hdr)          */
      v[15];       /* holding var for read-in EEG data          */



int i=0,           /* indicator for byte number of a value */
    ch=1,          /* channel counter                      */
    k,             /* counter for # time thru main drawing loop */
    s,             /* counter for # of EEG samples         */
    c,             /* variable used to chk for EOF         */
```

```
   minutes=0,     /* # minutes elapsed (for display)      */
   hardware,      /* type of hardware running             */
   origin_x, origin_y, /* Lower left corner of the window */
   size_x, size_y;     /* Size of the window */

float seconds=0.,
     data[18],                /* "volt" is a 15-element real number array  */
                              /*    used to hold the calculated voltage for*/
                              /*    each channel (0th element is not used) */
     max[19],     /* maximum value in each channel */
     mini[19],    /* minimum value in each channel */
     sum_of_sqs[15], /* sum of squared values in each channel */
     std_dev[19],    /* std deviation in each channel        */
     t=0.,           /* tracks elapsed time of display       */
     aspect;         /* Window's aspect ratio, x/y */

char sample_rate_char=0,  /* header value indicating record sample rate */
    *newfile1,    /* filename transfer variable  */
    *newfile2,    /* filename transfer variable  */
    xfer[25],     /* this worked (*xfer didn't) for sprint (see draw_status)*/
    time1[50];    /* elapsed time status line    */

Boolean noprep,   /* tracks whether prep_data() has been run */
        draw_active;  /* Is drawing active ? (a holdover from CURVE DEMO) */

Coord half, offset, text_offset;  /* length offsets for text */


/****  FUNCTION PROTOTYPES  ****/

void prep_data(void);            /* calls readhdr() and readch()  (see below) */
void readhdr(void);              /* reads in data from record header */
void readch(char cv[],char dc[],char sens,char caluv);
                                 /* reads data in counts and converts to volts*/
void time_print(void);           /* prints elapsed time to status box */
void init_window(char *argv[]);  /* initializes window */
void init_events(void);          /* defines valid actions for event queue */
void redraw(void);               /* redraws display when window is redrawn */
void process_left_down(void);    /* links event to action */
void process_left_up(void);      /* links event to action */
void process_middle_down(void);  /* links event to action */
void process_middle_up(void);    /* links event to action */
void draw_instructions(void);    /* draws help screen */
void quit(void);                 /* quits program gracefully */
void init_fonts(void);           /* defines text font */
void help(void);                 /* displays help screen */
void end_help(void);             /* removes help screen */

Boolean helping(void);           /* tracks presence of help screen */

extern void init_menus(void);    /* creates menus */
extern void draw_display(void); /* draws toposcope display */



/********************************** MAIN() ************************************/

void main(int argc, char *argv[])
{
        if (argc == 1)
        {
           printf("\n   USAGE:  topos infile.dat\n");
           quit();
        }

        strcpy(xfer,argv[1]);

        draw_active = FALSE;


        /********* initialize everything **********/

        init_window(argv);

        frontbuffer(TRUE);              /* Enables writing to front buffer   */
                                        /*  (Default is only back is enabled)*/

        czclear(0xcccccccc,0x7FFFFF);   /* Clears the viewport to lt gray  */
                                        /*  and sets the z-axis to minimum */
                                        /*  (0x7FFFFF) in enabled buffers  */

        frontbuffer(FALSE);             /* Disables writing to front buffer*/
                                        /*  (back buffer still enabled)    */

        init_fonts();

        init_menus();

        init_events();

        draw_instructions();

        while(TRUE)                     /* Continuously calls function which */
               event();                 /*  monitors for mouse actions, etc. */

}

/******************************** PREP_DATA() **********************************/


void prep_data(void)
{
```

```
noprep = FALSE;

if (fopen(xfer, "r") == NULL)
{
   printf("\n  Input file %s does not exist in this directory\n",xfer);
   quit();
}
else
   ifp = fopen(xfer, "r");


/************* create ".dat.chXX" files ***************/

for (ch=1;ch<=14;++ch)
{
    newfile1 = "MAXIMUMFILENAME.EXTENSION";
    newfile2 = "MAXIMUMFILENAME.EXTENSION";
    strcpy(newfile1,xfer);
    strcpy(newfile2,xfer);

    if (ch==1) strcat(newfile1,".ch1");
    if (ch==2) strcat(newfile1,".ch2");
    if (ch==3) strcat(newfile1,".ch3");
    if (ch==4) strcat(newfile1,".ch4");
    if (ch==5) strcat(newfile1,".ch5");
    if (ch==6) strcat(newfile1,".ch6");
    if (ch==7) strcat(newfile1,".ch7");
    if (ch==8) strcat(newfile1,".ch8");
    if (ch==9) strcat(newfile1,".ch9");
    if (ch==10) strcat(newfile1,".ch10");
    if (ch==11) strcat(newfile1,".ch11");
    if (ch==12) strcat(newfile1,".ch12");
    if (ch==13) strcat(newfile1,".ch13");
    if (ch==14) strcat(newfile1,".ch14");

    ofp[ch]=fopen(newfile1, "w+");

    max[ch]=0.;
    mini[ch]=0.;
}

strcat(newfile2,".h");
hfp=fopen(newfile2, "w+");


readhdr();

fseek(ifp,1920,0);           /* Sets the file pointer to start of  */
                             /* the EEG data (1920th byte) from the*/
                             /* beginning of the file (byte 0)]    */
    for (ch=1;ch<=14,++ch)
        sum_of_sqs[ch]=0.0;      /* Resets array used in Std Dev calc */

    k=1;

    while ((c = getc(ifp)) != EOF)
    {
        s=k-1;
        readch(cv, dc, sens, caluv);
        ++k;
    }

    for (ch=1;ch<=14;++ch)
        std_dev[ch] = sqrt(sum_of_sqs[ch]/k);

    for(ch=1;ch<=14;++ch)
        fprintf(hfp,"%f %f %i %f\n",max[ch],mini[ch],sample_rate_char,
                                                std_dev[ch]);

}


/******************************** READHDR() **********************************/

void readhdr(void)

{

   /*read in sampling rate of AD conversion*/

   fseek(ifp,416,0);
   fscanf(ifp,"%c",&sample_rate_char);

   if (sample_rate_char==7) sample_rate_char=128;
   if (sample_rate_char==15) sample_rate_char=64;


   /*read in calibration voltages*/

   for (ch=1;ch <= 14;++ch)
   {
        if (ch == 1)
          fseek(ifp,428,0);      /* Sets "ifp" to byte 428 from file start (0)*/

        if (ch == 3)
          fseek(ifp,1,1);        /* byte 431 (skipping 430) */
        if (ch == 7)
          fseek(ifp,1,1);        /* byte 436 (skipping 435) */

        if (ch == 11)            /* byte 441 (skipping 440) */
```

```
          fseek(ifp,1,1);

        if (ch == 14)           /* byte 445 (skipping 444) */
          fseek(ifp,1,1);

        fscanf(ifp, "%c", &AD);

        cv[ch]=AD;
   }


   /*read in dc offsets*/

   for (ch=1;ch <= 14;++ch)    {

        if (ch == 1)          /* Sets "ifp" to byte 449 from file start (0) */
          fseek(ifp,449,0);
        if (ch == 3)                 /* byte 452 (skipping 451) */
          fseek(ifp,1,1);

        if (ch == 7)                 /* byte 457 (skipping 456) */
          fseek(ifp,1,1);

        if (ch == 11)                /* byte 462 (skipping 461) */
          fseek(ifp,1,1);

        if (ch == 14)                /* byte 466 (skipping 465) */
          fseek(ifp,1,1);


        fscanf(ifp, "%c", &AD);

        dc[ch]=AD;

   }

   fscanf(ifp, "%c", &sens);
   fseek(ifp,1,1);                  /* byte 469 (skipping 468) */

   fscanf(ifp, "%c", &caluv);

}

/******************************* READCH() **********************************/

void readch(char cv[], char dc[], char sens, char caluv)
{                                    /* readin one sample of all 14 channels */
    fscanf(ifp,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",
        &bb,&bb,&v[1],&v[2],&bb,&v[3],&v[4],&v[5],&v[6],&bb,
        &v[7],&v[8],&v[9],&v[10],&bb,&v[11],&v[12],&v[13],&v[14],&bb);
                        /* Reads in block of 20 CHAR (1 byte) A-D values into */
                        /*  either the bit bucket variable or the CHAR array  */
                        /*  "v" according to the format found in the lab      */

    /* convert to voltages */
    for(ch=1;ch<=14;++ch)
    {
        data[ch] = ((float)v[ch]-(float)dc[ch])*(2.0*caluv/cv[ch])*(sens/10.0);

        if (data[ch] > max[ch])
                max[ch] = data[ch];

        if (data[ch] < mini[ch])
                mini[ch] = data[ch];

        /* write each channel's data to separate files */
         fprintf(ofp[ch],"%f\n",data[ch]);

        sum_of_sqs[ch] = sum_of_sqs[ch] + pow(data[ch],2);

    }

}


/******************************** INIT_WINDOW() ********************************/

/*
 * Opens the window and sets up the graphic mode.
 */
void init_window(char *argv[])
{
        char *t, *strrchr();
        int planes;

        keepaspect(.86,1);  /* x/y ratio of window */

        /* open the window with its invocation as the title (less the path) */
        winopen((t=strrchr(argv[0], '/')) != NULL ? t+1 : argv[0]);

        /*
         * If we run on an 8-bit eclipse, we don't use RGB mode or depth
         * cuing. Speed things up on all Eclipses.
         */
        planes = getplanes();

        if (planes <= 8)
                hardware = ECLIPSE8;
        else if (planes <= 24)
                hardware = ECLIPSE24;
        else
                hardware = GT;
        doublebuffer();

        if (hardware != ECLIPSE8)
                RGBmode();

        zbuffer(TRUE);
        gconfig();

        getorigin(&origin_x, &origin_y);
        getsize(&size_x, &size_y);

        aspect = (float)size_x/(float)size_y;
}


/****************************** INIT_EVENTS() ******************************/

/*
 * Tells event manager what to pay attention to.
 */
void init_events(void)
{
        /* ESC key and WINQUIT quit the program */
        add_event(ANY, ESCKEY, UP, quit, 0);
        qdevice(ESCKEY);
        add_event(ANY, WINQUIT, ANY, quit, 0);
        qdevice(WINQUIT);

        /* window redraw events */
        add_event(ANY, REDRAW, ANY, redraw, 0);
        qdevice(REDRAW);

        add_event(ANY,  LEFTMOUSE, DOWN, process_left_down, NULL);
        add_event(ANY,  LEFTMOUSE, UP, process_left_up, NULL);
        qdevice(LEFTMOUSE);

        add_event(ANY,  MIDDLEMOUSE, DOWN, process_middle_down, NULL);
        add_event(ANY,  MIDDLEMOUSE, UP, process_middle_up, NULL);
        qdevice(MIDDLEMOUSE);

        /* Draw the display whenever the user does not request anything */
        /* i.e., there are no events happening */
        add_update(&draw_active, draw_display, NULL);
}


/********************************* REDRAW() *******************************/

/*
 * Called by the event manager whenever a redraw event occurs,
 * e.g. when the window is resized.
 */
void redraw(void)
{
		getorigin(&origin_x, &origin_y);
		getsize(&size_x, &size_y);
		aspect = (float)size_x/(float)size_y;
		viewport(0, size_x-1, 0, size_y-1);

		if (helping())
			help();
		else
		{
			frontbuffer(TRUE);
			czclear(0xcccccccc,0x7FFFFF);
			frontbuffer(FALSE);

			draw_title();
			draw_boxes();
			draw_status();
		}
}

/*********************************OTHER FUNCTIONS*****************************/

void process_left_down(void)
{
	int marker;

	if(helping()){
		end_help();
		return;
	}
}

void process_left_up(void)
{
	unqdevice(MOUSEX);
	unqdevice(MOUSEY);
}

void process_middle_down(void)
{
	int marker;
	end_help();
}
```

```
void process_middle_up(void)
{
        unqdevice(MOUSEX);
        unqdevice(MOUSEY);
}


/* limits of the view volume */
Coord box_limit[3][2] =
        {       { -1.0, 1.0},
                { -1.0, 1.0},
                { -1.0, 1.0}
        };


void quit(void)
{
        gexit();
        fclose(ifp);

        for (ch=1;ch<=14;++ch)
           fclose(ofp[ch]);

        fclose(hfp);

        exit(0);
}




/*
 * Routines for writing the intro messages to the screen.
 */

double min(double x, double y)
{
        if (x < y)
                return x;
        else
                return y;
}

static fmfonthandle base_font, scaled_font;


void init_fonts(void)
{
        fminit();
        base_font = fmfindfont("Times-Roman");
}
```

```
void setup_string(char str[100], int size, Coord *half)
{
        double h_size, v_size, f_size;
        Coord length;

        h_size = (double)size_x/(double)size;
        v_size = (double)size_y/(double)size;
        f_size = min(h_size, v_size);
        scaled_font = fmscalefont(base_font, f_size);
        fmsetfont(scaled_font);

        length = fmgetstrwidth(scaled_font, str);
        *half = (Coord)(length/size_x);
}

static int looking_at_help;
static int active_state;

void help(void)
{
        if (!looking_at_help)
        {
                looking_at_help = TRUE;

                active_state = draw_active;
                draw_active = FALSE;
        }
        draw_instructions();
}

void end_help(void)
{
        if (looking_at_help)
        {
                looking_at_help = FALSE;
                draw_active = active_state;
        }
}

Boolean helping(void)
{
        return looking_at_help;
}

void time_print(void)
{
        ortho2(-1.0, 1.0, -1.0, 1.0);
        depthcue(FALSE);
        cpack(0xaa0000);

        minutes=(int)(t/60);
        seconds=t-((float)minutes*60);

        sprintf(time1, "Elapsed Time:  %i:%.3f sec", minutes,seconds);
        setup_string(time1, 70,&half);
        cmov2(.58,-0.80);
        fmprstr(time1);
}


void draw_instructions(void)
{
        Coord half, offset, text_offset;
        char str[100];

        ortho2(-1.0, 1.0, -1.0, 1.0);
        czclear(0xaaaaaaaa,0x7FFFFF);
        depthcue(FALSE);

        sprintf(str,"Topos");

        cpack(0x00555555);
        setup_string(str,10,&half);
        cmov2(0.005 - half, 0.705);
        fmprstr(str);

        /* set draw color to BLUE */
        if (hardware == ECLIPSE8)
                color(BLUE);
        else cpack(0xbb0000);

        setup_string(str,10,&half);
        cmov2(0.0 - half, 0.7);
        fmprstr(str);

        /* set draw color to BLUE */
        if (hardware == ECLIPSE8)
                color(BLUE);
        else
                cpack(0x880000);

        sprintf(str,"Interactive AFIT Toposcope");
        setup_string(str,30,&half);
        cmov2(0.0 - half, 0.5);
        fmprstr(str);

        /* set draw color to WHITE */
        if (hardware == ECLIPSE8)
                color(WHITE);
        else
                cpack(0x222222);

        /* just sets font size for body of help screen */
        setup_string(str,50,&half);

        offset = -0.6;
        text_offset = -0.30;

        sprintf(str,"Mouse Left:");
        cmov2(offset, 0.3);
        fmprstr(str);
        sprintf(str,"Halt display.");
        cmov2(text_offset, 0.3);
        fmprstr(str);
        sprintf(str,"(Requires \"Start Display\" from menu to restart.)");
        cmov2(text_offset, .22);
        fmprstr(str);

        sprintf(str,"Mouse Middle:");
        cmov2(offset, 0.10);
        fmprstr(str);
        sprintf(str,"Continuous Drawing - Press pauses, release restarts.");
        cmov2(text_offset, 0.10);
        fmprstr(str);

        sprintf(str,"Step Drawing - Press/release both draw next step.");
        cmov2(text_offset, .02);
        fmprstr(str);

        sprintf(str,"Batch Drawing - Press draws next batch (redraws all
           bins).");
        cmov2(text_offset, -.06);
        fmprstr(str);

        sprintf(str,"Mouse Right:");
        cmov2(offset, -.18);
        fmprstr(str);
        sprintf(str,"Main menu");
        cmov2(text_offset, -.18);
        fmprstr(str);

        swapbuffers();
}
```

## I.2 Screen Creation: tdraw.c

```
/*
 *
 * tdraw.c
 *
 * Part of "TOPOS" by Capt Dwight Roblyer, AFIT/GSO-92D
 * November 1992
 *
 * Based on "CURVE DEMO"
 * by Howard Look for Silicon Graphics, June 1989
 *
 * This file contains functions which draw the screen displays.
 *
 */


#include <stdio.h>
#include <gl.h>
#include <device.h>
#include <math.h>
#include "topos.h"
#include "tevent.h"


/* Prototypes */
void draw_display(void);      /* coordinates overall screen display */
void draw_boxes(void);        /* draws blue channel boxes */
void draw_bars(void);         /* draws bars displayed in each channel box */
void erase_bars(void);        /* overwrites bars with the background color */
void draw_status(void);       /* displays status and configuration info */
void draw_title(void);        /* draws title, electrode locations/labels/lines */
void draw_filename(void);     /* fills in status for data file name */
void draw_update(void);       /* fills in orig digitization rate of EEG file */
void draw_freq(void);         /* fills in instantaneous frequency of ref signal*/
void draw_matrix(void);       /* displays correlation matrix */
void draw_line(void);         /* line-drawing function */

extern void time_print();

int b,                    /* the ch selected as reference  */
    bb_integer,           /* bit-bucket for integer data types */
    bins,                 /* number of time-shifted bars before repeat */
    sample_rate,          /* record rate of digitized EEG record */
    lw,                   /* line or bar width (or height) */
    n,                    /* counter used for reading parameters for */
                          /*   hardcoded reference sinusoids */
    s_old,                /* sample number of most recent event */
    s_diff,               /* number of samples between two most recent events */
    shift,                /* tracks which time bin currently in use */
    shift_last,         /* time bin used last */
    pack=0,             /* used to define color of bars */
    s=0,                /* data sample currently being processed */
    cmt_min[10],        /* minute portion of data comment time-tag */
    cmt_sec[10],        /* seconds portion of data comment time-tag */
    cmt_sample[10]={0}, /* sample # at which to display each of 10 comments */
    cmt_cnt=0,          /* counter to track data file comments */
    flag,               /* counter for data file comment processing */
    buffer,             /* diagnostic: tracks display buffers */
    buffer_now;         /* diagnostic: holds current buffer */

float bb_float,              /* bit-bucket for decimal data types */
      packd=0.,              /* used to define color of bars */
      data_old[19]={0.},     /* data pt in each ch read last iteration */
      data_oldold[19]={0.}, /* data pt in each ch read two iterations ago */
      start[15][76][3]={0}, /* XYZ of start of bar by channel and shift */
      start_old[15][76][3]={0}, /* last value of "start" */
      stop[15][76][3]={0},      /* XYZ of bar stop by channel and shift */
      stop_old[15][76][3]={0},  /* last value of "stop"  */
      scale_factor[19]={0.},    /* 3 stand. dev. scale for each channel */
      correl[15]={0.},          /* channel correlations for matrix display */
      cursor_top[15][3],        /* coord of cursor's upper pt in each ch */
      cursor_bottom[15][3],     /* coord of cursor's lower pt in each ch */
      cursor_top_old[15][3],    /* last value of "cursor_top" */
      cursor_bottom_old[15][3], /* last value of "cursor_bottom */
      ypos[19]={0},             /* amplitude of each channel at time t */
      freq=0.,                 /* calculated, instant freq of reference signal */
      disp[19],                /* displacement of bar from x-axis by channel */
      pt1[3],pt2[3];           /* used for electrode-to-box lines */

FILE *tfp;              /* file ptr to datafile.DIAG output file */

Boolean cmt_flag[20]; /* holds which display comment has been displayed */

unsigned long packcolor[15];  /* special variable type used to hold */
                              /*  the "packed integer" required by  */
                              /*  the function "cpack"              */

char title[20], status[50], electrode[20], channel[10];
                              /* strings for drawing text to screen */
char cmt[20][20];             /* strings for drawing data file cmnts */

extern Boolean  time_shift,
                interval,
                axis_crossing,
                peak,
                constant_intensity,
                zero_ptrs,
                wait,
                single_buff,
                synch_constrained,
                batch;

extern int      hardware,
                total_skip,
                skip_seconds,
                ch,
                c,
                k,
                interval_skip;

extern float t,
             max[], mini[],
             std_dev[],
             data[],
             volt[],
             synch_limit;

extern char cv[],
            dc[],
            sens,
            caluv,
            xfer[],
            *newfile1,
            *newfile2;

extern FILE *ofp[],
            *hfp;

extern Booloan noprep,
               zero_pointers,
               synch_constrained;

extern Coord half;


/************************ FUNCTION "DRAW_DISPLAY" ***************************/

void draw_display(void)         /* Function used to draw the screen       */
{

        qreset();

        shift=0;

        frontbuffer(TRUE);        /* Enables front & back buffers for drawing */
        czclear(0xcccccccc,0x7FFFFF); /* Re-clears the viewport to gray and */
                                  /* sets the z-axis to minimum 0x7FFFFF      */

        setlinestyle(SOLID);
```

```
    frontbuffer(FALSE);     /* Disables front buffer for drawing */

    draw_title();

if (noprep)             /* If prep_data() was not run, program looks for */
                        /*  the .CHxx and .H files.  Reads in .H file.   */
{
      for (ch=1;ch<=14;++ch)
      {
            newfile1 = "MAXIMUMFILENAME.EXTENSION";
            newfile2 = "MAXIMUMFILENAME.EXTENSION";
            strcpy(newfile1,xfer);
            strcpy(newfile2,xfer);

            if (ch==1) strcat(newfile1,".ch1");
            if (ch==2) strcat(newfile1,".ch2");
            if (ch==3) strcat(newfile1,".ch3");
            if (ch==4) strcat(newfile1,".ch4");
            if (ch==5) strcat(newfile1,".ch5");
            if (ch==6) strcat(newfile1,".ch6");
            if (ch==7) strcat(newfile1,".ch7");
            if (ch==8) strcat(newfile1,".ch8");
            if (ch==9) strcat(newfile1,".ch9");
            if (ch==10) strcat(newfile1,".ch10");
            if (ch==11) strcat(newfile1,".ch11");
            if (ch==12) strcat(newfile1,".ch12");
            if (ch==13) strcat(newfile1,".ch13");
            if (ch==14) strcat(newfile1,".ch14");

            if (fopen(newfile1, "r") == NULL)
            {
               printf("\n  Input file %s does not exist in this
                                                         directory\n",
                      newfile1);
               quit();
            }
            else
               ofp[ch]=fopen(newfile1, "r");

            max[ch]=0.;
            mini[ch]=0.;
      }

    strcat(newfile2,".h");
    hfp=fopen(newfile2, "r");

    fseek(hfp,0,0);

    for (ch=1;ch<=14;++ch)
            fscanf(hfp,"%f%f%i%f",&max[ch],&mini[ch],&sample_rate,
                                                      &std_dev[ch]);
}

noprep=FALSE;          /* sets flag to not reopen files during session*/

ofp[0]=fopen("sinetest.dat.h","r");      /* opens data files for sine */
ofp[15]=fopen("sinetest.dat.1","r");     /*   reference signals       */
ofp[16]=fopen("sinetest.dat.2","r");
ofp[17]=fopen("sinetest.dat.1_5hz","r");
ofp[18]=fopen("sinetest.dat.halfhz","r");

for (n=15;n<=18;++n)
   fscanf(ofp[0],"%f%f%i%f",&max[n],&mini[n],&bb_integer,&std_dev[n]);

total_skip = skip_seconds*sample_rate;

if (zero_pointers)                    /* if starting @ beginning of data*/
            for (ch=1;ch<=14;++ch)
                     fseek(ofp[ch],0,0);

k=1;                          /* tracks # of times thru main draw loop  */
s_diff=0;                     /* tracks # of samples between draw event */
s_old=0;                      /* sample # of last draw event */



newfile1="MAXIMUMFILENAME.EXTENSION"; /* opens file for diagnostic */
strcpy(newfile1,xfer);                /*  output and prints hdr    */
strcat(newfile1,".diag");
tfp=fopen(newfile1,"w");

fprintf(tfp,"%s  (reference ch=%i)\n\n",xfer,n);
fprintf(tfp,"  s  SHIFT  buff  data[r] gray  1-data   1-disp   2-disp
                          3-disp   4-disp   5-disp   6-disp   7-disp\n");
fprintf(tfp,"                                                  8-disp   9-disp
                       10-disp  11-disp  12-disp  13-disp  14-disp\n\n");

buffer=2;



while ((c = getc(hfp)) != EOF)     /* reads in data file comments */
{
   fscanf(hfp,"%i%i%s",&cmt_min[cmt_cnt],&cmt_sec[cmt_cnt],cmt[cmt_cnt]);
   cmt_sample[cmt_cnt]=((cmt_min[cmt_cnt]*60)+cmt_sec[cmt_cnt])*64;
   ++cmt_cnt;
}

for (cmt_cnt=0;cmt_cnt<=9;++cmt_cnt)
```

```
   cmt_flag[cmt_cnt]=FALSE;


while ((c = getc(ofp[1])) != EOF)  /* begins main loop for screen u/d */
{

   while (s < total_skip)          /* bypasses data samples until      */
   {                               /*   selected start time is reached */
           ++s;
           for (ch=1;ch<=18;++ch)
                   fscanf(ofp[ch],"%f",&bb_float);
           frontbuffer(TRUE);

           if (s==1)
           {
           cpack(0x000000aa);
           sprintf(status, "SEARCHING");
           setup_string(status, 50,&half);
           cmov2(.625,-.62);
           fmprstr(status);
           cpack(0x777777);
           rect(.61,-.67,.87,-.74);
           cpack(0xaa);
           rectf(.62,-.68,.625,-.73);
           }

           if (s==(total_skip/8))
               rectf(.62,-.68,.65,-.73);
           if (s==(total_skip/4))
               rectf(.62,-.68,.68,-.73);
           if (s==(total_skip/8*3))
               rectf(.62,-.68,.71,-.73);
           if (s==(total_skip/2))
               rectf(.62,-.68,.74,-.73);
           if (s==(total_skip/8*5))
               rectf(.62,-.68,.77,-.73);
           if (s==(total_skip/4*3))
               rectf(.62,-.68,.80,-.73);
           if (s==(total_skip/8*7))
               rectf(.62,-.68,.83,-.73);
           if (s==(total_skip-4))
               rectf(.62,-.68,.86,-.73);
           frontbuffer(FALSE);
   }

   for (cmt_cnt=9;cmt_cnt>=0;--cmt_cnt)  /* chks if time for new data */
```

```
{                                        /*  comment                 */
   if (s>=cmt_sample[cmt_cnt])
   {
      if (cmt_flag[cmt_cnt]==FALSE)
      {
        frontbuffer(TRUE);
        cpack(0xcccccccc);
        rectf(-.17,.4,.17,.25);
      }

      break;
   }
}

if (cmt_flag[cmt_cnt]==FALSE)           /* draws data cmnt to screen */
{
        cpack(0x00ff0000);
        sprintf(status, "%s",cmt[cmt_cnt]);
        setup_string(cmt[cmt_cnt], 50,&half);
        cmov2(0.0 - half,.3);
        fmprstr(status);
        frontbuffer(FALSE);

        for (flag=cmt_cnt;flag>=0;--flag)
           cmt_flag[cmt_cnt]=TRUE;
}


frontbuffer(TRUE);        /*erases msg area over STATUS box*/
cpack(0xcccccccc);
rectf(.61,-.625,.95,-.525);
frontbuffer(FALSE);


if ((qtest() == 101 || qtest() == 103)  /* displays msg if rt or   */
{                                       /*  lft mouse pushed and   */
        frontbuffer(TRUE);              /*  stops draw loop        */
        cpack(0x000000aa);
        sprintf(status, "DISPLAY HALTED");
        setup_string(status, 50,&half);
        cmov2(.58,-.62);
        fmprstr(status);
        frontbuffer(FALSE);

        break;
}

if (qtest()==102 && !batch)      /* pauses display if middle mouse */
{
        qreset();
        frontbuffer(TRUE);
        cpack(0x000000aa);
        sprintf(status, "DISPLAY PAUSED");
        setup_string(status, 50,&half);
        cmov2(.58,-.62);
        fmprstr(status);
        frontbuffer(FALSE);

        while(qtest()!=102)
           ;
        qreset();
        frontbuffer(TRUE);
        cpack(0xcccccccc);
        rectf(.57,-.625,.92,-.525);
        frontbuffer(FALSE);

}

t=s*1.0/sample_rate;            /* time of sample from start of record*/

++s;                            /* increments "s"  for next pass */


for (ch=1;ch<=18;++ch)      /* saves most recent data values and */
        {                   /*   reads in new                    */
        data_oldold[ch] = data_old[ch];
        data_old[ch] = data[ch];

        fscanf(ofp[ch], "%f", &data[ch]);
        }

  if (interval)           /* enter if peak or axis-crossing only */
  {

    if (peak)             /* enter if peak-driven drawing */
    {
     if ( (data_old[b] >= data_oldold[b]) && (data[b] < data_old[b]))
     {                    /* chks for two increases then one drop */

      if ((k % interval_skip) == 0)  /* enter every Nth event as   */
                                     /*  set by Draw Interval menu */
      {

        if (!wait && !batch)         /* scales display spd to # bins*/
        {
           for (ch=1;ch<=((75-bins)*60000);++ch)
               ch=ch;
        }

        s_diff = s-s_old;
        s_old = s;
      swapbuffers();

      ++buffer;                    /* diagnostics for .diag   */
      buffer_now=(buffer % 2);

      erase_bars();

      draw_boxes();

      draw_status();

      draw_bars();

    }

   ++k;

  }
 }

 if (axis_crossing)   /* enter if axis-xings drive draw events */
 {
  if ( (data_old[b] <= 0.) && (data[b] > 0.) ||
                 (data_old[b] > 0.) && (data[b] <= 0.))
  {                  /* chks for up or downward crossings */


   if ((k % interval_skip) == 0)  /* enter every Nth event as   */
                                  /*  set by Draw Interval menu */
   {
     s_diff = s-s_old;
     s_old = s;

     if (!wait && !batch)         /* scales display spd to # bins*/
     {
        for (ch=1;ch<=((75-bins)*60000);++ch)
            ch=ch;
     }

     swapbuffers();

     ++buffer;                    /* diagnostics for .diag   */
     buffer_now=(buffer % 2);

     erase_bars();

     draw_boxes();
     draw_status();

     draw_bars();
```

```
                }

                ++k;



               }
              }
             }

             if (!interval)       /* enter if continuous drawing */
             {
                   swapbuffers();
                   erase_bars();
                   draw_boxes();
                   draw_status();
                   draw_bars();
             }


        }                         /* ends the main loop */

  while (!qtest() && time_shift)
          swapbuffers();




  }                               /* ends the function  */


  /**************************** FUNCTION "DRAW_BARS" *************************/

  void draw_bars()                  /*  used to draw the bar(s) in each box */
  {
    /* left and right edges of 14 channel boxes */
    static float left_edge[15]={   0.,   -.4,   -.7, -.45, -.55,
                                 -.6, -.35,   -.1,  .15,   .4,
                                 .35,  .25,    .5,   .2,  -.1};
    static float right_edge[15]={  0.,   -.2,   -.5, -.25, -.35,
                                -.4, -.15,    .1,  .35,   .6,
                                 .55,  .45,    .7,   .4,   .1};

    /* distance from window's x-axis to center of each channel box */
    static float vert_offset[15]={    0.,  .625,  .325,  .175, -.125,
                                 -.425, -.575, -.725, -.575, -.425,
                                 -.125,  .175,  .325,  .625,  .575};


    linewidth(lw);       /* expands the line segment to "lw" pixels wide  */
```

```
if (single_buff)        /* diagnostic for .diag print */
   frontbuffer(TRUE);

for (ch=1;ch<=18;++ch)  /* clips and scales bar's displace in each ch */
{
      if ((data_old[ch]>0.) && (data_old[ch]>(3.*std_dev[ch])))
              data_old[ch]=(3.* std_dev[ch]);

      if ((data_old[ch]<0.) && (data_old[ch]<(-3.*std_dev[ch])))
              data_old[ch]=(-3.* std_dev[ch]);

      scale_factor[ch]=.12/(3.* std_dev[ch]);

      disp[ch]=( data_old[ch] * scale_factor[ch] );

      ypos[ch] = disp[ch]+vert_offset[ch];

}

for (ch=1;ch<=14;++ch)  /* calcs start and stop of each "bar", keeping    */
{                       /* track of which bins they are in, if time-shftd */

   if (time_shift)
      {
      start[ch][shift][0]= (left_edge[ch]+0.0025+((float)shift*
                                                 0.2/(float)bins));
      start[ch][shift][1]= ypos[ch];
      start[ch][shift][2]= 0.;

      stop[ch][shift][0]= (start[ch][shift][0]-0.0025+(0.2/(float)bins));
      stop[ch][shift][1]= ypos[ch];
      stop[ch][shift][2]= 0.;
      }


   if (!time_shift)
      {
      start_old[ch][shift][0]=start[ch][shift][0]; /* xfers new to old     */
      start_old[ch][shift][1]=start[ch][shift][1]; /* (old needed to erase)*/
      start_old[ch][shift][2]=start[ch][shift][2];
      stop_old[ch][shift][0]=stop[ch][shift][0];
      stop_old[ch][shift][1]=stop[ch][shift][1];
      stop_old[ch][shift][2]=stop[ch][shift][2];

      start[ch][shift][0]= (left_edge[ch]+.005);
      start[ch][shift][1]= ypos[ch];
      start[ch][shift][2]= 0.;

      stop[ch][shift][0]= (right_edge[ch]+.005);
      stop[ch][shift][1]= ypos[ch];
```

```
    stop[ch][shift][2]= 0.;
    }

if (!constant_intensity && !synch_constrained)  /* set gradient bar color*/
{
    packd=(data_old[b]*(107./(3.* std_dev[b]))+147.);

    pack=packd;
    packcolor[ch]= pack;
    packcolor[ch]= (packcolor[ch] << 8) | pack;
    packcolor[ch]= (packcolor[ch] << 8) | pack;
    packcolor[ch]= (packcolor[ch] << 8) | pack;
                /* creates a "packed integer" consisting of 4 8-bit   */
                /*   integers which are the truncated results of      */
                /*   scaling the EEG Ch b voltage to a 255-step gray  */
                /*   scale.  All 4 integers are the same value to     */
                /*   create the gray scale by assigning identical vals*/
                /*   to the alpha, red, green, and blue components    */

    cpack(packcolor[ch]);          /* sets the draw color  */

}


if (synch_constrained)        /* set thresholded bar color */
{
    if ((disp[b] <= synch_limit) && (disp[b] >= -synch_limit))
        cpack(0x0);
    else
        cpack(0xaaaaaaaa);
}

if (constant_intensity)       /* set constant bar color  */
          cpack(0x0);



bgnline();        /* draws the bar using the arrays "start1" and "stop1"*/
      v3f(start[ch][shift]);
      v3f(stop[ch][shift]);
endline();

if (time_shift)        /* draws cursor if display is time-shifted */
{
    cursor_top_old[ch][0]=cursor_top[ch][0];
    cursor_top_old[ch][1]=cursor_top[ch][1];
    cursor_top_old[ch][2]=cursor_top[ch][2];
    cursor_bottom_old[ch][0]=cursor_bottom[ch][0];
    cursor_bottom_old[ch][1]=cursor_bottom[ch][1];
    cursor_bottom_old[ch][2]=cursor_bottom[ch][2];
```

```
      cursor_top[ch][0]=stop[ch][shift][0];
      cursor_top[ch][1]=vert_offset[ch]+.125;
      cursor_top[ch][2]=0.;
      cursor_bottom[ch][0]=cursor_top[ch][0];
      cursor_bottom[ch][1]=vert_offset[ch]+.11;
      cursor_bottom[ch][2]=0.;

      cpack(0x00ff0000);
      linewidth(1);

      bgnline();
       v3f(cursor_top[ch]);
       v3f(cursor_bottom[ch]);
      endline();

      linewidth(lw);              /*reset linewidth to bar value*/
  }

}


/* prints .diag file */

fprintf(tfp,"%4i  %2i    %i   %7.3f  %3i  %7.3f  %7.4f  %7.4f  %7.4f  %7.4f
        %7.4f  %7.4f  %7.4f %i:%f\n",(s-1),shift,buffer_now,data_old[b],pack,
         data_old[1],disp[1],disp[2],disp[3],disp[4],disp[5],disp[6],disp[7],
         ((s-1)/64/60),(((((float)s-1.)/64./60.)-((s-1)/64/60))*60.));
fprintf(tfp,"                                                  %7.4f  %7.4f  %7.4f
        %7.4f  %7.4f  %7.4f  %7.4f\n\n",disp[8],disp[9],
         disp[10],disp[11],disp[12],disp[13],disp[14]);


if (interval && bins<=24)     /* draws correllation grid */
   draw_matrix();

if (time_shift)
{
      if (shift <= (bins-2))  /* increments shift # */
              ++shift;
      else
      {
              shift = 0;

              if (batch && qtest()==102)      /* handles middle mouse in */
                  qreset();                   /*  step and batch modes   */
              while (batch && qtest()!=102)
              {
                 if (qtest()==101 || qtest()==103)
                     break;
              }
```

```
                if (batch && qtest()==102)
                    qreset();

        }
  }


  if (wait)            /* purges unwanted mouse events */
  {
      while (qtest()!=101 && qtest()!=102)
        {
        if (qtest()==103)
           qreset();
        }
      if (qtest()!=101)
        qreset();
  }

}                         /* ends the function   */


/*************************** FUNCTION "ERASE_BARS" ***************************/

void erase_bars()                /* used to erase the bars after they  */
                                 /*   have been moved to the back buffer */
{

  if (single_buff)          /* diagnostic for .diag print */
        frontbuffer(TRUE);


        cpack(0xcccccccc);   /* sets drawing color equal to backgrnd  */

        linewidth(lw); /* expands the line segment into a lw-pixel-wide bar */

  if (time_shift)
  {

        for (ch=1;ch<=14;++ch)
        {
          bgnline();         /* overwrites bar #ch in backgrnd color */
            v3f(start[ch][(shift)]);
            v3f(stop[ch][(shift)]);
          endline();


          frontbuffer(TRUE);
          linewidth(1);
          if (!single_buff)  /* overwrites last cursor if time-shifted */
          {
            bgnline();
              v3f(cursor_top_old[ch]);
              v3f(cursor_bottom_old[ch]);
            endline();
          }
          else
          {
            bgnline();
              v3f(cursor_top[ch]);
              v3f(cursor_bottom[ch]);
            endline();
          }
          linewidth(lw);
          if (!single_buff)
            frontbuffer(FALSE);

        }
    }

    if (!time_shift)
    {
          for (ch=1;ch<=14;++ch)
          {
          bgnline();         /* overwrites bar #ch in backgrnd color */
            v3f(start_old[ch][shift]);                    v3f(stop_old[ch][shift]);
          endline();
          }
    }

}                                 /* ends the function    */

/************************** FUNCTION "DRAW_MATRIX" ************************/

void draw_matrix(void)          /* draws correlation grid @ screen's center */
{
        static float xedge[6]={ -.15, -.09, -.03, .03, .09, .15};
        static float yedge[6]={ .025, -.05, -.125, -.2, -.275, -.35};
        static float xoff=.004;
        static float yoff=.005;
        float x1shift=0.;
        float x2shift=0.;

        frontbuffer(TRUE);


        linewidth(1);
        cpack(0x00ff0000);

        rect(xedge[1],yedge[0],xedge[2],yedge[1]);
        rect(xedge[2],yedge[0],xedge[3],yedge[1]);
        rect(xedge[3],yedge[0],xedge[4],yedge[1]);
rect(xedge[0],yedge[1],xedge[1],yedge[2]);
rect(xedge[1],yedge[1],xedge[2],yedge[2]);
rect(xedge[3],yedge[1],xedge[4],yedge[2]);
rect(xedge[4],yedge[1],xedge[5],yedge[2]);
rect(xedge[1],yedge[2],xedge[2],yedge[3]);
rect(xedge[3],yedge[2],xedge[4],yedge[3]);
rect(xedge[0],yedge[3],xedge[1],yedge[4]);
rect(xedge[1],yedge[3],xedge[2],yedge[4]);
rect(xedge[3],yedge[3],xedge[4],yedge[4]);
rect(xedge[4],yedge[3],xedge[5],yedge[4]);
rect(xedge[2],yedge[4],xedge[3],yedge[5]);

x1shift=shift*(.052/(float)bins);
x2shift=(shift+1)*(.052/(float)bins);

for (ch=1;ch<=14;++ch)
{
     correl[ch]=fabs(disp[ch]-disp[b]);

     if (correl[ch]<=(.12*.2))
        cpack(0x0);

     if (correl[ch]>(.12*.2) &&
                   correl[ch]<=((3.*std_dev[ch])*.3))
        cpack(0x00777777);

     if (correl[ch]>(.12*.3))
        cpack(0x00eeeeee);

     if (ch==1)
         rectf((xedge[1]+xoff+x1shift),(yedge[0]-yoff),
             (xedge[1]+xoff+x2shift),(yedge[1]+yoff));
     if (ch==14)
        rectf((xedge[2]+xoff+x1shift),(yedge[0]-yoff),
             (xedge[2]+xoff+x2shift),(yedge[1]+yoff));
     if (ch==13)
        rectf((xedge[3]+xoff+x1shift),(yedge[0]-yoff),
             (xedge[3]+xoff+x2shift),(yedge[1]+yoff));
     if (ch==2)
        rectf((xedge[0]+xoff+x1shift),(yedge[1]-yoff),
             (xedge[0]+xoff+x2shift),(yedge[2]+yoff));
     if (ch==3)
        rectf((xedge[1]+xoff+x1shift),(yedge[1]-yoff),
             (xedge[1]+xoff+x2shift),(yedge[2]+yoff));
     if (ch==11)
        rectf((xedge[3]+xoff+x1shift),(yedge[1]-yoff),
             (xedge[3]+xoff+x2shift),(yedge[2]+yoff));
     if (ch==12)
        rectf((xedge[4]+xoff+x1shift),(yedge[1]-yoff),
             (xedge[4]+xoff+x2shift),(yedge[2]+yoff));
     if (ch==4)
```

```
                rectf((xedge[1]+xoff+x1shift),(yedge[2]-yoff),
                     (xedge[1]+xoff+x2shift),(yedge[3]+yoff));
             if (ch==10)
                rectf((xedge[3]+xoff+x1shift),(yedge[2]-yoff),
                     (xedge[3]+xoff+x2shift),(yedge[3]+yoff));
             if (ch==5)
                rectf((xedge[0]+xoff+x1shift),(yedge[3]-yoff),
                     (xedge[0]+xoff+x2shift),(yedge[4]+yoff));
             if (ch==6)
                rectf((xedge[1]+xoff+x1shift),(yedge[3]-yoff),
                     (xedge[1]+xoff+x2shift),(yedge[4]+yoff));
             if (ch==8)
                rectf((xedge[3]+xoff+x1shift),(yedge[3]-yoff),
                     (xedge[3]+xoff+x2shift),(yedge[4]+yoff));
             if (ch==9)
                rectf((xedge[4]+xoff+x1shift),(yedge[3]-yoff),
                     (xedge[4]+xoff+x2shift),(yedge[4]+yoff));
             if (ch==7)
                rectf((xedge[2]+xoff+x1shift),(yedge[4]-yoff),
                     (xedge[2]+xoff+x2shift),(yedge[5]+yoff));


        }



        frontbuffer(FALSE);
}


/*************************** FUNCTION "DRAW_BOXES" **************************/

void draw_boxes(void)               /* draws boxes on screen for ch displays  */
{

        linewidth(1);

        cpack(0x00ff0000);          /* sets line color to blue  */

        rect(-.4,.75,-.2,.625);          /* CH #1  */
        rect(-.4,.625,-.2,.5);

        sprintf(channel, "CH 1");
        setup_string(channel, 60,&half);
        cmov2(-.39,.51);
        fmprstr(channel);

        rect(-.7,.45,-.5,.325);   /* CH #2  */
        rect(-.7,.325,-.5,.2);
```

```
sprintf(channel, "CH 2");
setup_string(channel, 60,&half);
cmov2(-.69,.21);
fmprstr(channel);

rect(-.45,.3,-.25,.175);  /* CH #3  */
rect(-.45,.175,-.25,.05);

sprintf(channel, "CH 3");
setup_string(channel, 60,&half);
cmov2(-.44,.06);
fmprstr(channel);

rect(-.55,0.,-.35,-.125); /* CH #4  */
rect(-.55,-.125,-.35,-.25);

sprintf(channel, "CH 4");
setup_string(channel, 60,&half);
cmov2(-.54,-.24);
fmprstr(channel);

rect(-.6,-.3,-.4,-.425);  /* CH #5  */
rect(-.6,-.425,-.4,-.55);

sprintf(channel, "CH 5");
setup_string(channel, 60,&half);
cmov2(-.59,-.54);
fmprstr(channel);

rect(-.35,-.45,-.15,-.575); /* CH #6  */
rect(-.35,-.575,-.15,-.7);

sprintf(channel, "CH 6");
setup_string(channel, 60,&half);
cmov2(-.34,-.69);
fmprstr(channel);

rect(-.1,-.6,.1,-.725);          /* CH #7  */
rect(-.1,-.725,.1,-.85);

sprintf(channel, "CH 7");
setup_string(channel, 60,&half);
cmov2(-.09,-.84);
fmprstr(channel);

rect(.15,-.45,.35,-.575); /* CH #8  */
rect(.15,-.575,.35,-.7);

sprintf(channel, "CH 8");
setup_string(channel, 60,&half);
cmov2(.16,-.69);
```

```
    fmprstr(channel);

    rect(.4,-.3,.6,-.425);          /* CH #9  */
    rect(.4,-.425,.6,-.55);

    sprintf(channel, "CH 9");
    setup_string(channel, 60,&half);
    cmov2(.41,-.54);
    fmprstr(channel);

    rect(.35,0.,.55,-.125);          /* CH #10  */
    rect(.35,-.125,.55,-.25);

    sprintf(channel, "CH 10");
    setup_string(channel, 60,&half);
    cmov2(.36,-.24);
    fmprstr(channel);

    rect(.25,.3,.45,.175);          /* CH #11  */
    rect(.25,.175,.45,.05);

    sprintf(channel, "CH 11");
    setup_string(channel, 60,&half);
    cmov2(.26,.06);
    fmprstr(channel);

    rect(.5,.45,.7,.325);          /* CH #12  */
    rect(.5,.325,.7,.2);

    sprintf(channel, "CH 12");
    setup_string(channel, 60,&half);
    cmov2(.51,.21);
    fmprstr(channel);

    rect(.2,.75,.4,.625);          /* CH #13  */
    rect(.2,.625,.4,.5);

    sprintf(channel, "CH 13");
    setup_string(channel, 60,&half);
    cmov2(.21,.51);
    fmprstr(channel);

    rect(-.1,.7,.1,.575);          /* CH #14  */
    rect(-.1,.575,.1,.45);

    sprintf(channel, "CH 14");
    setup_string(channel, 60,&half);
    cmov2(-.09,.46);
    fmprstr(channel);

}                             /* end of function     */
```

```
/***************************** FUNCTION "DRAW_STATUS" ************************/

void draw_status(void)
{
        cpack(0x00eeeeee);          /* draws Status box in near-white */
        rectf(.55,-.65,.95,-.95);

        cpack(0x00eeeeee);          /* draws Configuration box in same */
        rectf(-.96,-.65,-.55,-.95);

        time_print();               /* draws elapsed time */

        cpack(0x00aa0000);          /* sets color to dk red */


        /* remainder draws information into display boxes */

        sprintf(status, "STATUS");
        setup_string(status, 50,&half);
        cmov2(.68,-.70);
        fmprstr(status);

        draw_filename();
        draw_update();
        draw_freq();

        cpack(0x00aa0000);

        sprintf(status, "CONFIGURATION");
        setup_string(status, 50,&half);
        cmov2(-.925,-.70);
        fmprstr(status);

        cpack(0x00aa0000);

        if (b<=14)
                sprintf(status, "Reference Ch:       %i", b);
        if (b==15)
                sprintf(status, "Reference Ch:   1-Hz SINE");
        if (b==16)
                sprintf(status, "Reference Ch:   2-Hz SINE");
        if (b==17)
                sprintf(status, "Reference Ch:   1.5-Hz SINE");
        if (b==18)
                sprintf(status, "Reference Ch:   0.5-Hz SINE");
        setup_string(status, 70,&half)
        cmov2(-.93,-.75);
        fmprstr(status);
```

```
if ((interval) && (peak))
        sprintf(status, "Draw Event:   Peak");
if ((interval) && (axis_crossing))
        sprintf(status, "Draw Event:   Axis Xing");
if (!interval)
        sprintf(status, "Draw Event:   Constant");
setup_string(status, 70,&half);
cmov2(-.93,-.80);
fmprstr(status);

if (interval)
        sprintf(status, "  -> Display Interval:   %i",interval_skip)
setup_string(status, 70,&half);
cmov2(-.93,-.85);
fmprstr(status);

if (wait)
{
        cpack(0xaa);

        sprintf(status, "STEP-DRAW");
        setup_string(status, 50,&half);
        cmov2(-.86,-.58);
        fmprstr(status);

        sprintf(status, "(Middle Mouse = NEXT)");
        setup_string(status, 70,&half);
        cmov2(-.89,-.62);
        fmprstr(status);

        cpack(0x00ff0000);
}

if (batch)
{
        cpack(0xaa);

        sprintf(status, "BATCH-DRAW");
        setup_string(status, 50,&half);
        cmov2(-.89,-.58);
        fmprstr(status);

        sprintf(status, "(Middle Mouse = NEXT)");
        setup_string(status, 70,&half);
        cmov2(-.91,-.62);
        fmprstr(status);

        cpack(0x00ff0000);
}

if (constant_intensity)
```

THIS

PAGE

IS

MISSING

IN

ORIGINAL

DOCUMENT

pg. 139

```
                sprintf(status, "Ref Ch Freq:   %4.2f Hz", freq);
        else
                sprintf(status, "Ref Ch Freq:   N/A");
        setup_string(status, 70,&half);
        cmov2(.58,-.90);
        fmprstr(status);


}

/************************** FUNCTION "DRAW_TITLE" *************************/

void draw_title(void)     /* draws title, electrodes, etc */
{
        frontbuffer(TRUE);

        ortho2(-1.0, 1.0, -1.0, 1.0);
        depthcue(FALSE);

        cpack(0x00777777);

        sprintf(title, "TOPOS");
        setup_string(title, 15,&half);
        cmov2(-.925,.805);
        fmprstr(title);

        cpack(0xaa);
        cmov2(-.93,.80);
        fmprstr(title);



        cpack(0x77);

        sprintf(title, "The AFIT");
        setup_string(title, 40,&half);
        cmov2(-.93,.70);
        fmprstr(title);

        sprintf(title, "Interactive");
        setup_string(title, 40,&half);
        cmov2(-.93,.63);
        fmprstr(title);

        sprintf(title, "Toposcope");
        setup_string(title, 40,&half);
        cmov2(-.93,.56);
        fmprstr(title);

        cpack(0x777777);
```

```
sprintf(title, "Face");
setup_string(title, 45,&half);
cmov2(0.0-half,.95);
fmprstr(title);

sprintf(title, "Right");
setup_string(title, 45,&half);
cmov2(.86,0.);
fmprstr(title);

sprintf(title, "Ear");
setup_string(title, 45,&half);
cmov2(.88,-.07);
fmprstr(title);

sprintf(title, "Left");
setup_string(title, 45,&half);
cmov2(-.95,0.);
fmprstr(title);

sprintf(title, "Ear");
setup_string(title, 45,&half);
cmov2(-.945,-.07);
fmprstr(title);


circf(-.15,.625,.01);                      /*  electrode: f3  */
circf(-.5,.6,.01);                      /*  electrode: f7  */
circf(-.75,.125,.01);                      /*  electrode: t3  */
circf(-.15,.125,.01);                      /*  electrode: c3  */
circf(-.7,-.35,.01);                      /*  electrode: t5  */
circf(-.25,-.35,.01);                      /*  electrode: p3  */
circf(-.25,-.775,.01);                      /*  electrode: o1  */
circf(.25,-.775,.01);                      /*  electrode: o2  */
circf(.25,-.35,.01);                      /*  electrode: p4  */
circf(.7,-.35,.01);                      /*  electrode: t6  */
circf(.15,.125,.01);                      /*  electrode: c4  */
circf(.75,.125,.01);                      /*  electrode: t4  */
circf(.5,.6,.01);                      /*  electrode: f8  */
circf(.15,.625,.01);                      /*  electrode: f4  */

sprintf(electrode, "F3");
setup_string(electrode, 60,&half);
cmov2(-.165,.645);
fmprstr(electrode);

sprintf(electrode, "F7");
setup_string(electrode, 60,&half);
cmov2(-.515,.62);
fmprstr(electrode);
```

```
sprintf(electrode, "T3");
setup_string(electrode, 60,&half);
cmov2(-.765,.075);
fmprstr(electrode);

sprintf(electrode, "C3");
setup_string(electrode, 60,&half);
cmov2(-.165,.145);
fmprstr(electrode);

sprintf(electrode, "T5");
setup_string(electrode, 60,&half);
cmov2(-.715,-.4);
fmprstr(electrode);

sprintf(electrode, "P3");
setup_string(electrode, 60,&half);
cmov2(-.265,-.33);
fmprstr(electrode);

sprintf(electrode, "O1");
setup_string(electrode, 60,&half);
cmov2(-.265,-.825);
fmprstr(electrode);

sprintf(electrode, "O2");
setup_string(electrode, 60,&half);
cmov2(.235,-.825);
fmprstr(electrode);

sprintf(electrode, "P4");
setup_string(electrode, 60,&half);
cmov2(.235,-.33);
fmprstr(electrode);

sprintf(electrode, "T6");
setup_string(electrode, 60,&half);
cmov2(.685,-.4);
fmprstr(electrode);

sprintf(electrode, "C4");
setup_string(electrode, 60,&half);
cmov2(.135,.145);
fmprstr(electrode);

sprintf(electrode, "T4");
setup_string(electrode, 60,&half);
cmov2(.735,.075);
fmprstr(electrode);

sprintf(electrode, "F8");
```

```
setup_string(electrode, 60,&half);
cmov2(.485,.62);
fmprstr(electrode);

sprintf(electrode, "F4");
setup_string(electrode, 60,&half);
cmov2(.135,.645);
fmprstr(electrode);

linewidth(2);

arc(0.,.85,.08,-100,1900);
arc(.25,.77,.06,200,1600);
arc(-.25,.77,.06,200,1600);

deflinestyle(1,0xcccc);
setlinestyle(DOTTED);

pt1[0]= -.1;
pt1[1]= .625;
pt2[0]= -.2;
pt2[1]= .625;
draw_line();

pt1[0]= -.4;
pt1[1]= .625;
pt2[0]= -.5;
pt2[1]= .6;
draw_line();

pt1[0]= -.5;
pt1[1]= .6;
pt2[0]= -.6;
pt2[1]= .45;
draw_line();

pt1[0]= -.7;
pt1[1]= .25;
pt2[0]= -.75;
pt2[1]= .125;
draw_line();

pt1[0]= -.75;
pt1[1]= .125;
pt2[0]= -.45;
pt2[1]= .125;
draw_line();

pt1[0]= -.25;
pt1[1]= .125;
pt2[0]= -.15;
```

```
pt2[1]= .125;
draw_line();

pt1[0]= -.15;
pt1[1]= .125;
pt2[0]= -.35;
pt2[1]= -.05;
draw_line();

pt1[0]= -.55;
pt1[1]= -.2;
pt2[0]= -.7;
pt2[1]= -.35;
draw_line();

pt1[0]= -.7;
pt1[1]= -.35;
pt2[0]= -.6;
pt2[1]= -.35;
draw_line();

pt1[0]= -.4;
pt1[1]= -.35;
pt2[0]= -.25;
pt2[1]= -.35;
draw_line();

pt1[0]= -.25;
pt1[1]= -.35;
pt2[0]= -.25;
pt2[1]= -.45;
draw_line();

pt1[0]= -.25;
pt1[1]= -.7;
pt2[0]= -.25;
pt2[1]= -.775;
draw_line();

pt1[0]= -.25;
pt1[1]= -.775;
pt2[0]= -.1;
pt2[1]= -.775;
draw_line();

pt1[0]= .1;
pt1[1]= -.775;
pt2[0]= .25;
pt2[1]= -.775;
draw_line();
```

```
pt1[0]= .25;
pt1[1]= -.775;
pt2[0]= .25;
pt2[1]= -.7;
draw_line();

pt1[0]= .25;
pt1[1]= -.45;
pt2[0]= .25;
pt2[1]= -.35;
draw_line();

pt1[0]= .25;
pt1[1]= -.35;
pt2[0]= .4;
pt2[1]= -.35;
draw_line();

pt1[0]= .6;
pt1[1]= -.35;
pt2[0]= .7;
pt2[1]= -.35;
draw_line();

pt1[0]= .7;
pt1[1]= -.35;
pt2[0]= .55;
pt2[1]= -.2;
draw_line();

pt1[0]= .35;
pt1[1]= -.05;
pt2[0]= .15;
pt2[1]= .125;
draw_line();

pt1[0]= .15;
pt1[1]= .125;
pt2[0]= .25;
pt2[1]= .125;
draw_line();

pt1[0]= .45;
pt1[1]= .125;
pt2[0]= .75;
pt2[1]= .125;
draw_line();

pt1[0]= .75;
pt1[1]= .125;
pt2[0]= .7;
```

```
        pt2[1]= .25;
        draw_line();

        pt1[0]= .6;
        pt1[1]= .45;
        pt2[0]= .5;
        pt2[1]= .6;
        draw_line();

        pt1[0]= .5;
        pt1[1]= .6;
        pt2[0]= .4;
        pt2[1]= .625;
        draw_line();

        pt1[0]= .2;
        pt1[1]= .625;
        pt2[0]= .1;
        pt2[1]= .625;
        draw_line();

        setlinestyle(SOLID);

        frontbuffer(FALSE);

}

/**************************** FUNCTION "DRAW_LINE" **************************/

void draw_line(void)      /* used immediately above by draw_title() */
{
        bgnline();
           v3f(pt1);
           v3f(pt2);
        endline();
}
```

## I.3 User Menus: tmenu.c

```
/*
 *
 * tmenu.c
 *
 * Part of "TOPOS" by Capt Dwight Roblyer, AFIT/GSO-92D
 * November 1992
 *
 * Based on "CURVE DEMO"
 * by Howard Look for Silicon Graphics, June 1989
 *
 * This file contains the routines used for initializing and
 * maintaining the menus that make up the profile program.
 *
 */


#include <stdio.h>
#include <gl.h>
#include <device.h>
#include <math.h>
#include "tevent.h"
#include "topos.h"

/* Prototypes */
void init_menus(void);          /* initializes menu */
void do_menus(void);            /* calls menus */
void remake_curve_menu(void);   /* builds a menu */
void remake_display_menu(void);
void remake_interval_menu(void);
void remake_bin_menu(void);
void remake_start_menu(void);
void remake_timer_menu(void);
void remake_synch_menu(void);
void remake_line_menu(void);
void remake_intensity_menu(void);
void remake_all_menus(void);

extern void prep_data(void);
extern void set_b(int);
extern void set_lw(int);
extern void set_intensity(int);
extern void set_bins(int);
extern void set_defaults(void);
extern void change_display(int);
extern void set_interval_skip(int);
extern void leave_ptrs(void);
extern void zero_ptrs(void);
extern void position_ptrs(int);
extern void save_quit(void);
extern void quit(void);
extern void invert_order(void);
extern void help(void);
extern void end_help(void);
extern void swap(void);
extern void set_test(int);
extern void draw_display(void);

extern char xfer[];

/* Declare menus */
static long curve_menu;                  /* Top level Menu */
static long display_menu;
static long interval_menu;
static long bin_menu;
static long start_menu;
static long timer_menu;
static long synch_menu;
static long line_menu;
static long intensity_menu;
static long test_menu;


/************************* FUNCTION: INIT_MENUS() *****************************/
/* Allocate and initialize the menus, then tell the event manager
 * to watch for right mouse button clicks.
 */
void init_menus(void)
{
        curve_menu = newpup();
        display_menu = newpup();
        interval_menu = newpup();
        bin_menu = newpup();
        start_menu = newpup();
        timer_menu = newpup();
        scale_menu = newpup();
        synch_menu = newpup();
        line_menu = newpup();
        intensity_menu = newpup();
        test_menu = newpup();
        set_initial_conditions();
        remake_all_menus();

        add_event(ANY,  RIGHTMOUSE, DOWN, do_menus, NULL);
        qdevice(RIGHTMOUSE);
}

/************************** FUNCTION: DO_MENUS() *****************************/
/*
```

```
 * Called by the event manager whenever the right mouse button is
 * pressed.
 */
void do_menus(void)
{
        end_help();
        dopup(curve_menu);
}


/*********************** FUNCTION: REMAKE_CURVE_MENU() **********************/

void remake_curve_menu(void)
{
        freepup(curve_menu);
        curve_menu = newpup();

        addtopup(curve_menu, "TOPOS %t"); /* title */
        addtopup(curve_menu, "Prepare Data %f", prep_data);
        addtopup(curve_menu, "Reference CH %m", synch_menu);
        addtopup(curve_menu, "Bar Width %m", line_menu);
        addtopup(curve_menu, "Bar Intensity %m", intensity_menu);
        addtopup(curve_menu, "Display Mode %m", display_menu);
        addtopup(curve_menu, "  Draw Interval %m", interval_menu);
        addtopup(curve_menu, "  Bin Number %m", bin_menu);
        addtopup(curve_menu, "Start Display %m", start_menu);
        addtopup(curve_menu, "Help %f", help);
        addtopup(curve_menu, "Test %m", test_menu);
        addtopup(curve_menu, "Swap Buffers %f", swap);
        addtopup(curve_menu, "Quit %f", quit);

}


/********************* FUNCTION: REMAKE_DISPLAY_MENU() **********************/

void remake_display_menu(void)
{
        char menu_string[32];

        freepup(display_menu);
        display_menu = newpup();

        addtopup(display_menu, "Display Mode %t %F", change_display);
        addtopup(display_menu, "Single Bar & Continuous %x0");
        addtopup(display_menu, "Single Bar & Interval (axis xing) %x1");
        addtopup(display_menu, "Single Bar & Interval (peak) %x2");
        addtopup(display_menu, "Time-Shift & Continuous %x3");
        addtopup(display_menu, "Time-Shift & Interval (axis xing) %x4");
        addtopup(display_menu, "Time-Shift & Interval (peak) %x5");
}
```

```
/********************* FUNCTION: REMAKE_INTERVAL_MENU() *********************/

void remake_interval_menu(void)
{
        char menu_string[32];

        freepup(interval_menu);
        interval_menu = newpup();

        addtopup(interval_menu, "Set Display Interval %t %F",
                                                      set_interval_skip);
        addtopup(interval_menu, "   Every 2nd event %x2");
        addtopup(interval_menu, "   Every 3rd event %x3");
        addtopup(interval_menu, "   Every 4th event %x4");
        addtopup(interval_menu, "   Every 5th event %x5");
        addtopup(interval_menu, "   Every 6th event %x6");
        addtopup(interval_menu, "   Every 7th event %x7");
        addtopup(interval_menu, "   Every 8th event %x8");
        addtopup(interval_menu, "   Every 9th event %x9");
        addtopup(interval_menu, "   Every 10th event %x10");
        addtopup(interval_menu, "   All events (Reset) %x1");
}

/************************ FUNCTION: REMAKE_BIN_MENU() ************************/

void remake_bin_menu(void)
{
        char menu_string[32];

        freepup(bin_menu);
        bin_menu = newpup();

        addtopup(bin_menu, "Set # of Time-Shift Bins %t %F",
                                                      set_bins);
        addtopup(bin_menu, "   6 %x6");
        addtopup(bin_menu, "   10 %x10");
        addtopup(bin_menu, "   24 %x24");
        addtopup(bin_menu, "   50 %x50");
        addtopup(bin_menu, "   74 %x74");
}

/*********************** FUNCTION: REMAKE_START_MENU() ***********************/

void remake_start_menu(void)
{
        char menu_string[32];
        freepup(start_menu);
        start_menu = newpup();

        addtopup(start_menu, "Start Display at... %t");
        addtopup(start_menu, "Beginning of Data %f", zero_ptrs);
        addtopup(start_menu, "Current Position %f", leave_ptrs);
        addtopup(start_menu, "Selected Time  %m", timer_menu);
}


/********************** FUNCTION: REMAKE_TIMER_MENU() **********************/

void remake_timer_menu(void)
{
        char menu_string[32];

        freepup(timer_menu);
        timer_menu = newpup();

        addtopup(timer_menu, "Elapsed Time (in mins:secs) %t %F",
                                                   position_ptrs);
        addtopup(timer_menu, "      1 %x60");
        addtopup(timer_menu, "      2 %x120");
        addtopup(timer_menu, "      3 %x180");
        addtopup(timer_menu, "      4 %x240");
        addtopup(timer_menu, "      4:45 %x285");
        addtopup(timer_menu, "      5 %x300");
        addtopup(timer_menu, "      6 %x360");
        addtopup(timer_menu, "      8 %x480");
        addtopup(timer_menu, "     10 %x600");
        addtopup(timer_menu, "     12 %x720");
}

/********************** FUNCTION: REMAKE_SYNCH_MENU() **********************/

void remake_synch_menu(void)
{
        freepup(synch_menu);
        synch_menu = newpup();

        addtopup(synch_menu, "Reference Channel %t %F", set_b);
        addtopup(synch_menu, "     CH 1 %x1"):
        addtopup(synch_menu, "     CH 2 %x2");
        addtopup(synch_menu, "     CH 3 %x3");
        addtopup(synch_menu, "     CH 4 %x4");
        addtopup(synch_menu, "     CH 5 %x5");
        addtopup(synch_menu, "     CH 6 %x6");
        addtopup(synch_menu, "     CH 7 %x7");
        addtopup(synch_menu, "     CH 8 %x8");
        addtopup(synch_menu, "     CH 9 %x9");
        addtopup(synch_menu, "     CH 10 %x10");
        addtopup(synch_menu, "     CH 11 %x11");
        addtopup(synch_menu, "     CH 12 %x12");
        addtopup(synch_menu, "     CH 13 %x13");
        addtopup(synch_menu, "     CH 14 %x14");
        addtopup(synch_menu, "   0.5 Hz Sine %x18");
        addtopup(synch_menu, "   1   Hz Sine %x15");
        addtopup(synch_menu, "   1.5 Hz Sine %x17");
        addtopup(synch_menu, "   2   Hz Sine %x16");
}

/********************** FUNCTION: REMAKE_LINE_MENU() **********************/

void remake_line_menu(void)
{
        freepup(line_menu);
        line_menu = newpup();

        addtopup(line_menu, "Pixel Width %t %F", set_lw);
        addtopup(line_menu, "     2 %x2");
        addtopup(line_menu, "     5 %x5");
        addtopup(line_menu, "    10 %x10");
        addtopup(line_menu, "    15 %x15");
        addtopup(line_menu, "    20 %x20");
        addtopup(line_menu, "    25 %x25");
        addtopup(line_menu, "    30 %x30");
}

/******    ************** FUNCTION: REMAKE_INTENSITY_MENU() ********************/

void remake_intensity_menu(void)
{
        freepup(intensity_menu);
        intensity_menu = newpup();

        addtopup(intensity_menu, "Bar Intensity %t %F", set_intensity);
        addtopup(intensity_menu, "Scale & Clip @ 3 Sigmas %x0");
        addtopup(intensity_menu, "Clamp to Maximum %x1");
        addtopup(intensity_menu, "Threshold @ Reference < 0.005 %x2");
        addtopup(intensity_menu, "Threshold @ Reference < 0.01 %x3");
        addtopup(intensity_menu, "Threshold @ Reference < 0.02 %x4");

}

/*********************** FUNCTION: REMAKE_TEST_MENU() **********************/

void remake_test_menu(void)
{
        freepup(test_menu);
        test_menu = newpup();

        addtopup(test_menu, "Test Modes %t %F", set_test);
        addtopup(test_menu, "Step Draw ON %x1");
        addtopup(test_menu, "Step Draw OFF %x2");
        addtopup(test_menu, "Batch Draw ON %x5");
        addtopup(test_menu, "Batch Draw OFF %x6");
        addtopup(test_menu, "Single Buffer ON %x3");
        addtopup(test_menu, "Single Buffer OFF %x4");
        addtopup(test_menu, "TEST MODE OFF %x0");
}

/*********************** FUNCTION: REMAKE_ALL_MENUS() ***********************/

void remake_all_menus()
{
        remake_display_menu();
        remake_interval_menu();
        remake_bin_menu();
        remake_timer_menu();
        remake_start_menu();
        remake_synch_menu();
        remake_line_menu();
        remake_intensity_menu();
        remake_test_menu();
        remake_curve_menu();
}
```

## I.4 Display Option Control: tcontrol.c

```
/*
 *
 * tcontrol.c
 *
 * Part of "TOPOS" by Capt Dwight Roblyer, AFIT/GSO-92D
 * November 1992
 *
 * Based on "CURVE DEMO"
 * by Howard Look for Silicon Graphics, June 1989
 *
 * This file contains the routines used for the program's display and
 * environment options, as defaults and as selected by the user. Most
 * of these routines are called by menu selection. See the file tmenu.c
 * for menu definitions.
 *
 */


#include <stdio.h>
#include <gl.h>
#include <device.h>
#include <math.h>
#include "topos.h"
#include "tevent.h"


/* Prototypes */
void set_initial_conditions(void);
void set_defaults(void);
void change_display(int);
void change_scale(int);
void set_interval_skip(int);
void set_bins(int);
void set_b(int);
void set_lw(int);
void set_intensity(int);
void leave_ptrs(void);
void zero_ptrs(void);
void position_ptrs(int);
void swap(void);
void dbl_synch(void);
void set_test(int);

extern void clear_window(void);
extern void remake_curve_menu(void);
extern void remake_display_menu(void);
extern void remake_all_menus(void);
```

```
int skip_minutes=0,
    skip_seconds=0,
    total_skip=0,
    interval_skip;
    line_style;

float synch_limit;

Boolean time_shift,
        interval,
        axis_crossing,
        peak,
        constant_intensity,
        zero_pointers,
        wait,
        single_buff,
        synch_constrained,
        batch;

extern Boolean draw_active, noprep;
extern char tool;
extern float myscale;
extern int hardware,
           sample_rate,
           s,
           bins,
           b,
           lw,
           ch;
extern FILE *ofp[];

/* About the window... */
extern int origin_x, origin_y, size_x, size_y;
extern float aspect;

/************************FUNCTION: SET_INITIAL_CONDITIONS()******************/
/*
 * Sets default values for the TOPOS options. THOSE WHICH USERS
 * MIGHT WANT TO CHANGE ARE MARKED.
 */

void set_initial_conditions(void)
{
        ortho(-1.0, 1.0, -1.0, 1.0, -2.0, 2.0);
        line_style = SOLID;
        setlinestyle(line_style);
        unqdevice(INPUTCHANGE);

        noprep = TRUE;
        interval_skip=1;            /* which draw events to display */

        lw=25;                      /* pixel height of bar */

        time_shift = TRUE;          /* time_shifted or single bar */
        bins=6;                     /* # of bins in time-shifted display*/

        interval = TRUE;            /* continuous or interval drawing */
        axis_crossing = TRUE;       /*   if interval, then axis-xing   */
        peak = FALSE;               /*   if interval, then peaks       */

        skip_minutes = 0;           /* starts at beginning of data file */

        b=17;                     /* reference channel */
                                  /* ch15=1hz ch16=2hz ch17=1.5hz ch18=.5hz */

        single_buff=TRUE;         /* single or double screen buffers */

        wait=FALSE;               /* step-draw if TRUE */
        batch=TRUE;               /* batch-draw if TRUE */

        constant_intensity = TRUE;  /* bar's color is constant or scaled */
        synch_constrained=FALSE;    /* bar's color is thresholded */
        synch_limit=0.01;           /*   threshold values for colors */
}

/************************FUNCTION: SET_DEFAULTS()****************************/

void set_defaults(void)
{
        set_initial_conditions();
        remake_all_menus();
        draw_display();
}

/************************FUNCTION: CHANGE_DISPLAY()**************************/

void change_display(int new_mode)    /* implements Display Mode menu option */
{
        if (new_mode==0)
        {
                time_shift = FALSE;
                interval = FALSE;
                axis_crossing = FALSE;
                peak = FALSE;
        }

        if (new_mode==1)
        {
                time_shift = FALSE;
                interval = TRUE;
```

```
		axis_crossing = TRUE;
		peak = FALSE;
	}

	if (new_mode==2)
	{
		time_shift = FALSE;
		interval = TRUE;
		axis_crossing = FALSE;
		peak = TRUE;
	}

	if (new_mode==3)
	{
		time_shift = TRUE;
		interval = FALSE;
		axis_crossing = FALSE;
		peak = FALSE;
	}

	if (new_mode==4)
	{
		time_shift = TRUE;
		interval = TRUE;
		axis_crossing = TRUE;
		peak = FALSE;
	}

	if (new_mode==5)
	{
		time_shift = TRUE;
		interval = TRUE;
		axis_crossing = FALSE;
		peak = TRUE;
	}

	remake_display_menu();
	remake_curve_menu();
}

/************************FUNCTION: SET_INTERVAL_SKIP()**********************/

void set_interval_skip(int skip)   /* implements Draw Interval menu option */
{
	if (skip)
		interval_skip = skip;
	else
		interval_skip = 1;
}
```

THIS

PAGE

IS

MISSING

IN

ORIGINAL

DOCUMENT

pg. 158

```
        if (intensity==4)
        {
            constant_intensity=FALSE;
            synch_constrained=TRUE;
            synch_limit=0.02;
        }

}


/************************FUNCTION: ZERO_PTRS()****************************/

void zero_ptrs(void)   /* implements Start Display (Beginning) menu option */
{
        s=0;

        zero_pointers=TRUE;

        skip_seconds = 0;

        draw_display();
}

/************************FUNCTION: LEAVE_PTRS()****************************/

void leave_ptrs(void)  /* implements Start Display (Current) menu option */
{
        zero_pointers=FALSE;

        skip_seconds = 0;

        draw_display();
}

/***********************FUNCTION: POSITION_PTRS()*************************/

void position_ptrs(int skip)  /* implements Start Display (@) menu option */
{
        zero_pointers=TRUE;

        s=0;

        skip_seconds = skip;

        draw_display();
}
```

```
/*****************************FUNCTION: SWAP()******************************/

void swap(void)              /* implements Swap Buffers menu option */
{
        while(!qtest())
                swapbuffers();
}


/**************************FUNCTION: SET_TEST()***************************/

void set_test(int mode)      /* implements Test Modes menu option */
{
        if (mode==0)
        {
           wait=FALSE;
           single_buff=FALSE;
           batch=FALSE;
        }

        if (mode==1)
        {
           wait=TRUE;
           batch=FALSE;
        }

        if (mode==2)
        {
           wait=FALSE;
           batch=FALSE;
        }

        if (mode==3)
           single_buff=TRUE;

        if (mode==4)
           single_buff=FALSE;

        if (mode==5)
        {
           batch=TRUE;
           wait=FALSE;
        }

        if (mode==6)
        {
           batch=FALSE;
           wait=FALSE;
        }
}
```

## I.5 Event Queue Control: tevent.c

NOTE: *The following code was modified very little during this research.*

```
/*
 * tevent.c
 *
 * Part of "TOPOS" by Capt Dwight Roblyer, AFIT/GSO-92D
 * November 1992
 *
 * Used with few changes from "CURVE DEMO"
 * by Howard Look for Silicon Graphics, June 1989
 *
 * A more rational way to handle reading the event queue
 * Written by Wade Olsen
 *
 */

#include <stdio.h>
#include <device.h>
#include "tevent.h"


typedef struct event_s
{
    int window, device, state;
        void (*func)(void *, int);
        char *arg;
    struct event_s        *next;
} event_t, *event_p;

typedef struct update_s
{
    int *flag;
        void (*ufunc)(void *);
        char *arg;
    struct update_s *next;
} update_t, *update_p;


static event_p     event_list;
static update_p    update_list;

/*
 * This routine adds an event to be checked for to the event queue.
 * window should be the wid of the window to respond in, or ANY if
 * this event applies to all windows.  device is the device, and state
 * is the device's value (e.g.  ANY, UP, DOWN, the window id (for
 * REDRAW), etc).  Func is the function that will be called, with
 * arguments 'arg' and the device's value.
 *
 * NOTE:  the device must be queued for it to be found by the event()
 * routine-- add_event DOES NOT qdevice(device).
 */
void
add_event(window, device, state, func, arg)
int   window, device, state;
void (*func)(void *, int);
char *arg;
{
    event_p new_guy;

            new_guy = (event_p)malloc(sizeof(event_t));
    new_guy->window = window;
    new_guy->device = device;
    new_guy->state  = state;
    new_guy->func   = func;
    new_guy->arg    = arg;
    new_guy->next   = event_list;
    event_list = new_guy;
}

/*
 *       Specify a function to be called if there is nothing in the queue
 * and (*flag) is non-zero.  If no update function is active, or
 * (*flag) is 0, then event() will block on a qread.  If there is an
 * active update function, event() will continuously call the update
 * function, hogging the cpu.
 */
void
add_update(flag, ufunc, arg)
int        *flag;
void (*ufunc)(void *);
char *arg;
{
    update_p        new_guy;

    new_guy = (update_p)malloc(sizeof(update_t));
    new_guy->flag = flag;
    new_guy->ufunc = ufunc;
        new_guy->arg  = arg;
    new_guy->next = update_list;
    update_list = new_guy;
}

/*
```

```
 * The main Event.  Call this repeatedly to automagically handle
 * reading the queue, and calling your functions to handle what
 * appears there.
 */
void
event()
{
        void find_event(void), event_inputchange(void);
        int find_update(void);
        static int initialized = 0;

        if (initialized == 0)
        {
                add_event(ANY, INPUTCHANGE, ANY, event_inputchange, NULL);
                initialized = 1;
        }

        /* Something in the queue?  Handle it */
        if (qtest())
                find_event();
}

int
find_update()
{
    update_p        scan;
    int                updated = 0;

    for (scan = update_list; scan && updated == 0; scan = scan->next)
        {
                if (*scan->flag)
                {
                    (*scan->ufunc)(scan->arg);
                    updated = 1;
                }
        }

    return(updated);
}

int context, state, device;

void
event_inputchange()
{
        context = winget();
}

void
find_event()
{
    event_p scan;
        short   s;

    device = qread(&s);
        state = s;
    for (scan = event_list; scan; scan = scan->next)
        {
                if ( ((scan->window == ANY) || (context == scan->window))
                        && ((scan->device == ANY) ||(device == scan->device))
                        && ((scan->state == ANY) || (state == scan->state)))
                {
                    (*scan->func)(scan->arg, state);
                }
        }
}
```

```
/*
 *
 * topos.h
 * This file contains the defines used in TOPOS.
 *
 */

/* Linestyles */
#define SOLID       0
#define DOTTED      1
#define DASHED      2

/* Patterns for the linestyles... */
/* 2 pixels on, 2 pixels off */
#define DOTTED_PATTERN 0xCCCC
/* 8 pixels on, 8 pixels off */
#define DASHED_PATTERN 0xFF00

#define ON      1
#define OFF     0

/* Hardware types */
#define ECLIPSE8        1
#define ECLIPSE24       2
#define GT              3

*****************************************************************************
*****************************************************************************

/*
 *       event.h
 * External interface and defines to input-queue event handling
 * routines.
 * Written by Wade Olsen for Silicon Graphics, Inc.
 */

/*
 *       The event handler understands two kinds of things; events and
 * updates.  Events are reactions to things occuring in the input
 * queue.  Updates are functions that should be called whenever there
 * is nothing waiting in the input queue, and may be active or
 * inactive.  If there are no active updates and nothing in the input
 * queue, then event() will block, using up no CPU time.
 *
 * add_event is used to look for events.  The first three arguments
 * are used to identify which event to look for.  The first argument
 * is the window (gid) the event must happen in; if this value is ANY
 * then any window will do.  The second argument is the device to look
 * for (e.g.  RIGHTMOUSE or REDRAW or KEYBD, etc).  Again, if it is
 * ANY then any device will match.  The third argument is the value
 * the device must generate (e.g.  DOWN or UP); ANY means all values
 * match.  The last two arguments are what should be done when an
 * event is generated.  The fourth argument is a function to be
 * called, and the fifth is an argument that should be supplied to the
 * function.  In addition, the value generated by the device will also
 * be passed to the function when it is called.
 *
 * For example,
 *        add_event(winget(), RIGHTMOUSE, DOWN, dopup, my_menus);
 *        qdevice(RIGHTMOUSE);
 * will make a pop-up menu appear when the right mousebutton goes
 * down.  Note that you must do the qdevice() call yourself.
 */
void add_event(int, int, int, void (*fn)(void *, int), char *) ;

/*
 * An update is like an event, only simpler.  The first argument is a
 * pointer to an integer flag specifying whether or not this update
 * function is active.  The second is a function to be called when it
 * is active, and the last is an argument to be supplied to the
 * function.
 */
void add_update(int *, void (*fn)(void *), char *) ;

/*
 * Finally, when all updates and events have been added, repeatedly
 * call event() to handle them -- something like
 *
 *        while (quitflag == FALSE) event();
 *
 * You should have previously added an event that sets quitflag to
 * TRUE, of course.
 */
void event(void) ;

/*
 *        These are some useful defines for the possible values buttons
 * can generate.
 */
#define ANY      -1
#define UP        0
#define DOWN      1

/*
 *        And a few external variables you might find useful
 */
extern int context, state, device;
```

## *I.7 Makefile*

```
#
# This makefile creates the test AFIT toposcopeo, TOPOS
#

LLDLIBS = -lfm_s -lgl_s -lm

CFILES = topos.o tevent.o tmenu.o tdraw.o tcontrol.o


topos: ${CFILES}
 <TAB>  cc -g -o topos ${CFILES} ${LLDLIBS}
```

## *Bibliography*

1. Badeau, Maj Albert F. Deputy Chief, Performance Assessment and Interface Technology Branch. Personal interview. Armstrong Medical Laboratory, Wright-Patterson AFB OH, May through July 1992.
2. Banducci, Capt Todd M. *An Analysis of the Effects of Phenytoin in Treating Motion Sickness and the Effects of Motion Sickness on the Human Electroencephalogram.* MS Thesis, AFIT/GSO/ENG/90D-2. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1990. (AD-A230423)
3. *Brain Atlas EEG Data File Structure.* GMP #820.181-5.2. Bio-logic Systems Corporation, Mundelein IL, December 1990.
4. *Brain Atlas User Manual.* Version 2.30. Bio-logic Systems Corporation, Mundelein IL, December 1989.
5. Chelen, William *et al.* "Use of Phenytoin in the Prevention of Motion Sickness," *Aerospace, Space, and Environmental Medicine 61:* 1022–1025 (November 1990).
6. Cooper, R. *et al. EEG Technology.* London: Butterworth and Company, 1974.
7. Coppola, Richard. "Issues in Topographic Analysis of EEG Activity," *Topographic Mapping of Brain Electrical Activity,* edited by Frank H. Duffy. Boston: Butterworth Publishers, 1986.
8. Davis, Jeffrey P., *et al.* "Space Motion Sickness During 24 Flights of the Space Shuttle," *Aerospace, Space, and Environmental Medicine, 59:* 1185–1189 (December 1988).
9. Dhenin, Geoffrey. *Aviation Medicine: Physiology and Factors.* London: Tri-Med Books Limited, 1978.
10. Duffy, Frank H. *et al.* "Significance Probability Mapping: An aid in the Topographic Analysis of Brain Electrical Activity," *Electroencephalography and Clnical Neurophysiology, 51:* 455–462 (May 1981).
11. Duffy, Frank H. "Topographic Mapping of Brain Electrical Activity: Clinical Applications and Issues," *Topographic Brain Mapping of EEG and Evoked Potentials,* edited by K. Maurer. New York: Springer-Verlag, 1989.
12. Gevins, Alan S. 'Overview of Computer Analysis," *Handbook of Electroencephalography and Clinical Neurophysiology,* Volume 1, edited by A.S. Gevins and A. Rémond. New York: Elsevier Science Publishers, 1987.
13. Harner, Richard N. "Clinical Application of Computed EEG Topography," *Topographic Mapping of Brain Electrical Activity,* edited by Frank H. Duffy. Boston: Butterworth Publishers, 1986.

14. Hettinger, Lawrence J. *et al.* "Motion and Human Performance," *Motion and Space Sickness*, edited by George H. Crampton. Boca Raton, FL: CRC Press, 1990.

15. Huntoon, C. *Human Tolerance to Space Flight*, "AIAA/NASA Symposium on the Maintainability of Aerospace Systems, 26-27 July 1989." Washington: AIAA, 1989 (AIAA-89-5062).

16. Institute of Medicine. *Mapping the Brain and its Functions*. Washington D.C.: National Academy Press, 1991.

17. John, E.R. *et al.* "Neurometric Topographic Mapping EEG avoked Potential Features: Application to Clinical Diagnosis and Cognitive Evalauation," *Topographic Brain Mapping of EEG and Evoked Potentials*, edited by K. Maurer. New York: Springer-Verlag, 1989.

18. Jones, David R. *et al.* "Self-Control of Psychophysiologic Response to Motion Stress: Using Biofeedback to Treat Airsickness," *Aerospace, Space, and Environmental Medicine 56:* 1152-1157 (December 1985).

19. Kellogg, Dean L. *Vestibular System Physiology and space Motion Sickness: An Introduction*. Final report, 1 May 1985-31 May 1985, to the USAF School of Aerospace Medicine. San Antonio TX: University of Texas Health Science Center, August 1986.

20. Lehmann, Deitrich. "From Mapping to the Analysis and Interpretation of EEG/EP Maps," *Topographic Brain Mapping of EEG and Evoked Potentials*, edited by K. Maurer. New York: Springer-Verlag, 1989.

21. Lehmann, Deitrich. "Principles of Spatial Analysis," *Handbook of Electroencephalography and Clinical Neurophysiology*, Volume 1, edited by A.S. Gevins and A. Rémond. New York: Elsevier Science Publishers, 1987.

22. Lehmann, Deitrich. "Spatial Anaysis of EEG and Evoked Potential Data," *Topographic Mapping of Brain Electrical Activity*, edited by Frank H. Duffy. Boston: Butterworth Publishers, 1986.

23. Lesévre, Nicole and Antoine Rémond. "Selected Applications of a Topographic Approach to Event-Relaied Potentials," *Topographic Mapping of Brain Electrical Activity*, edited by Frank H. Duffy. Boston: Butterworth Publishers, 1986.

24. Lopes da Silva, F.H. "A Critical Review of Clinical Applications of Topographic Mapping of Brain Potentials," *Journal of Clinical Neurophysiology*, 7: 535-551 (1990).

25. National Academy of Sciences: Committee on Space Biology and Medicine, *A Strategy for Space Biology and Medical Science for the 1980s and 1990s*. Washington: National Academy Press, 1987 (N89-24024)

26. Neidermeyer, Ernst and Fernando Lopes da Silva. *Electroencephalography*. Baltimore: Urban and Schwarzenberg, 1981

27. Nuwer, Marc R. "EEG Topographic Mapping and Frequency Analysis: Techniques and Studies in Clinical Setting," *Seminars in Neurology, 10:* 166–177 (June 1990).

28. Oman, Charles M. *et al.* "Space Motion Sickness Monitoring Experiment: Spacelab 1," *Motion and Space Sickness*, edited by George H. Crampton. Boca Raton FL: CRC Press, 1990.

29. Petsche, H. "From Graphein to Topos: Past and Future of Brain Mapping," *Topographic Brain Mapping of EEG and Evoked Potentials*, edited by K. Maurer. New York: Springer-Verlag, 1989.

30. Pingree, B.J.W. "Space Motion Sickness," *Journal of the Royal Navy Medical Service, 76:* 25–32 (Spring 1990).

31. Prichep, L.S. and E.R. John. "Neurometrics: Clinical Applications," *Handbook of Electroencephalography and Clinical Neurophysiology*, Volume 2, edited by F.H. Lopes da Silva *et al.* New York: Elsevier Science Publishers, 1987.

32. Ratino, D.A. *et al.* "Quantification of Reaction Time and Time Perception During Space Shuttle Operations," *Aerospace, Space, and Environmental Medicine 59:* 220–224 (March 1988).

33. Reason, J.T. and J.J. Brand. *Motion Sickness*. New York: Academic Press, 1975.

34. Rémond, Antoine. Preface to *Handbook of Electroencephalography and Clinical Neurophysiology*, Volume 1, edited by A.S. Gevins and A. Rémond. New York: Elsevier Science Publishers, 1987.

35. Rémond, Antoine. "Topological Aspects of the Organization, Processing, and Presentation of Data," *Symposium on the Analysis of Central Nervous System and Cardiovascular Data Using Computer Methods*. 73–93. Washington D.C.: National Aeronautics and Space Administration, 1965.

36. Sanger, David E. "A Japanese Innovation: The Space Antihero," *The New York Times, 140:* 1+ (December 8, 1990).

37. Shipton, Harold W. and Gerald L. Armstrong. "A Modern Frequency and Phase Indicating Toposcope," *Electroencephalography and Clinical Neurophysiology, 52:* 659–662 (December 1981).

38. Shipton, Harold W. "A New Frequency- Selective Toposcope for Electroencephalography, *Medical Electronics and Biology Engineering", 1:* 483–495 (1963).

39. Shipton, Harold W. "From Entertainment to Education, from Education to Enlightenment," *Topographic Mapping of Brain Electrical Activity*, edited by Frank H. Duffy. Boston: Butterworth Publishers, 1986.

40. Silicon Graph .s, Inc. *C Language Reference Manual*. Version 1.0. Document Number 007-0701-030. Mountain View, CA: Silicon Graphics, Inc., 1987.

41. Spehlmann, R. *EEG Primer.* New York: Elsevier Biomedical Press, 1988.

42. Thornton, William E. *et al.* "Clinical Characterization and Etiology of Space Motion Sickness," *Aerospace, Space, and Environmental Medicine 58:* A1–A8 (September 1987).

43. Tou, Julius . and Rafael C. Gonzalez. *Pattern Recognition Principles.* Reading, MS: Addison-Wesley Publishing Company, 1974.

44. Vogen, Capt George S. *A Topographical Analysis of the Human Electroencephalogram for Patterns in the Development of Motion Sickness.* MS Thesis, AFIT/GSO/ENG/91D-17. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991 (AD-A243656).

45. Walter, W. Grey and H.W. Shipton. "A New Toposcopic Display System," *Electroencephalography and Clinical Neurophysiology, 3*: 281–292 (1951).

46. Walter, W. Grey. "The Electrical Activity of the Brain," *Scientific American, 190*: 54–63 (June 1954).

47. Walter, W. Grey. *The Living Brain.* New York: W.W. Norton and Company, 1953.

48. Wong, Peter K.H. *Introduction to Brain Topography.* New York: Plenum Press, 1991.

## *Vita*

Captain Dwight A. Roblyer was born on 8 March 1962 in Neosho, Missouri. He graduated from Sam Houston High School in Arlington, Texas in May 1980 and entered Texas A&M University and the Aggie Corps of Cadets. In May 1984 he graduated with a Bachelor of Arts in Physics and was commissioned a second lieutenant in the United States Air Force. Captain Roblyer's first duty assignment was as a member of the cadre chosen to transition the Defense Satellite Communication System, Phase II (DSCS II), from Air Force Systems Command and contractor operators to the active-duty mission control crews of the new Air Force Space Command. This task began in AFSCF/VX at Sunnyvale Air Force Station, California, where he was certified as a ground systems controller. In February 1987 he moved to Falcon Air Force Base, Colorado, where he certified as a mission controller and flight commander and helped to activate the 3d Satellite Control Squadron to operate DSCS II and three other Department of Defense communication satellite constellations. There, he was selected to be among the unit's first group of evaluators and became the deputy chief of the standardization/evaluation branch overseeing AFSPACECOM's largest mission-ready crew force. In May 1991, Capt Roblyer was assigned to the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, to pursue a Master of Science degree in Space Operations.

Permanent address: 2101 Elaine Ct.
Arlington, Texas 76010

| REPORT DOCUMENTATION PAGE | Form Approved [illegible] |
|---|---|

[illegible]

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE December 1992 | 3. REPORT TYPE AND DATES COVERED Master's Thesis |

| | |
|---|---|
| 4. TITLE AND SUBTITLE A TOPOSCOPIC INVESTIGATION OF BRAIN ELECTRICAL ACTIVITY INDUCED BY MOTION SICKNESS | 5. FUNDING NUMBERS |
| 6. AUTHOR(S) Dwight A. Roblyer Captain, USAF | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583 | 8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSO/ENG/92D-03 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
| 11. SUPPLEMENTARY NOTES | |
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited | 12b. DISTRIBUTION CODE |

13. ABSTRACT *(Maximum 200 words)*

A toposcope was constructed as a new tool to study signal frequency and phase relationships in electroencephalogram (EEG) records collected while subjects were experiencing motion sickness. This new tool, named TOPOS, is a software-based, multi-featured version of Grey Walter and Harold Shipton's device which they first produced in the late 1940s. The TOPOS graphical display permits the study of instantaneous frequency relationships between the input channels and a reference signal of fixed or varying frequency. TOPOS adds a correlation grid to aid observers in detecting channel-to-reference correlation levels. Users can also vary several display parameters via menus to optimize the analysis environment. Sinusoidal test inputs of known frequency produced recognizable and predictable patterns on the TOPOS display, depending on the existing channel-to-reference frequency relationships. Motion-sickness-affected EEG was input to TOPOS in order to study the correlation between the displays of each channel and four separate references: a 1.5-Hz sinusoid, and three channels of the EEG itself. Rapidly varying correlations were observed in each case.

| | | | |
|---|---|---|---|
| 14. SUBJECT TERMS Toposcope, Brain Topographic Mapping, Motion Sickness, Electroencephalograph | | | 15. NUMBER OF PAGES 184 |
| | | | 16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED | 20. LIMITATION OF ABSTRACT UL |